IP Geolocation

by

Ovidiu Dan

A Dissertation
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Computer Science

Lehigh University
May 2019

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Ovidiu Dan
IP Geolocation

Date

Prof. Brian D. Davison, Dissertation Director, Chair
(Must Sign with Blue Ink)

Accepted Date

Committee Members

Prof. Yinzhi Cao

Prof. Michael F. Spear

Prof. Ting Wang

Dr. Fred Douglis

Acknowledgments

I would like to express my sincere gratitude to Prof. Brian D. Davison for his continuous guidance throughout the years. His timely suggestions have always helped put me on the right track. I would not have been able to finish this degree without his help.

I would also like to thank the rest of my thesis committee: Prof. Yinzhi Cao, Prof. Michael F. Spear, Prof. Ting Wang, and Dr. Fred Douglis for their support and thoughtful questions. I am grateful to Dr. Douglis for his detailed suggestions on improving the dissertation draft.

I am indebted to my managers at Microsoft: Vaibhav Parikh, Marcelo De Barros, Kumar Srinivasamurthy, and Manish Mittal for allowing me to pursue a PhD while working full time, and Siddhartha Arora and his colleagues in the User Location team for introducing me to the problem of IP geolocation and granting me access to their computing resources to perform experiments.

My sincere thanks also go to my mentors: Dr Mary Fernández for believing in me even before we met in person, Eddy de Rooij for helping me prepare for graduate school, and Drs. Pavel Dmitriev, Junlan Feng, Bernard Renger, and Ryen W. White for introducing me to research in an industry setting.

Last but not least, words cannot express how grateful I am to my family and friends: my wife Diana and daughter Sophie for putting up with the many times when I could not be with them during weekends and evenings; my father, mother, and brother for helping me get to where I am today, and Horatiu for being an outstanding friend.

To Diana and Sophie

Contents

List of Tables

List of Figures

Abstract

IP Geolocation databases map IP addresses to their corresponding geographical locations. They are used to find the approximate location of user's IP address at city level granularity, when the exact user location is not available. These databases are crucial to a variety of online services, including content personalization, credit card fraud prevention, geographic traffic load balancing, and location-based content licensing. Several companies offer commercial IP geolocation services. However, these commercial databases can be inaccurate and the methods they employ are proprietary. Furthermore, previous academic work in this area has typically resulted in low accuracy, it has focused on small geographic regions, and it has used small ground truth datasets. Consequently, investigating multiple geolocation approaches and evaluating their accuracy on a larger scale is worthwhile.

The overarching goal of this dissertation is to compile a geolocation database from scratch, by combining IP location information derived from multiple data sources using novel techniques. We develop seven IP geolocation approaches using a variety of data sources such as application logs, reverse DNS hostnames, traceroute paths, and information extracted from WHOIS databases. We place particular emphasis on search engine logs, as they have not been previously used to improve IP geolocation. From these logs we derive location information from user queries, user clicks, and from opt-in GPS data. Finally, we combine the output of all seven approaches into a single geolocation database. Since two or more approaches can provide conflicting location information for the same network block, we need to resolve location conflicts. We cast this conflation task as a machine learning problem. We train a classifier which for a given IP range can decide which location candidate is most likely correct.

We evaluate the resulting geolocation database using a ground truth set of 70 million IP addresses with known location, the largest ever reported in literature. We show that with a median error of only 3.59 kilometers, it significantly outperforms current state of the art commercial and academic approaches.

Chapter 1

Introduction

1.1 IP Geolocation and Its Importance

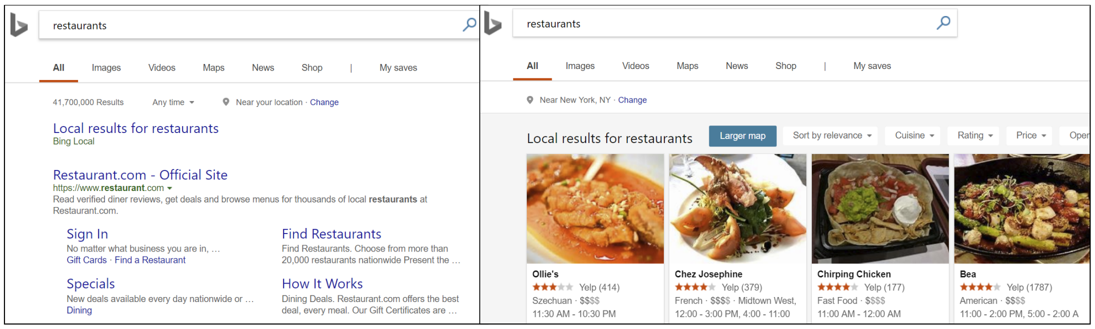
IP Geolocation databases map IP addresses to their corresponding geographical locations. They are used to find the approximate location of an IP address at the city level. Records in these databases contain IP ranges along with their physical location. Table 1.1 lists a few examples of such records. For instance, the second example in the table maps a block of 256 IP addresses to Guangdong, a city in China.

Table 1.1: Example of entries from an IP Geolocation database.

StartIP	EndIP	Country	State	City
1.0.16.0	1.0.16.255	JP	Tokyo	Tokyo
124.173.109.0	124.173.109.255	CN	Guangdong	Guangzhou
187.153.184.0	187.153.184.255	MX	Quintana Roo	Cancun

These databases are vital to a variety of online services when the exact location of a user is not available. Web search engines are an example of services which rely on this type of information. They use the geographical location of users to personalize the results shown on the page. For instance, for the query "weather" search engines may display a page block with the local weather forecast based on the location of the user.

Figure 1.1 shows another example of location-based personalization in the



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 …

Local results for restaurants

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

Figure 1.1: Effect of missing location information on search engine local search personalization. The left image displays the results for the query "restaurants" when the location of the user is unknown, while the right image displays the personalized experience for a user located in New York City.

context of search engines. The figures demonstrate the striking difference in results for the query "restaurants" when the user location is unknown, compared to when it is known. The generic nationwide results require the user to requery for more specific restaurants in their area, while the personalized results directly list restaurants tailored to a specific location.

Outside of the realm of search engines, IP geolocation services are used for applications such as:

- **Content Delivery Networks** deploy servers in datacenters across the globe to speed up access to content. IP geolocation databases help direct users to the closest server. This can result in improvements in latency, throughput, and bandwidth costs [1, 2].

- **Credit card fraud protection**: Financial institutions use the location of users making online transactions as one of the inputs in their credit card fraud detection algorithms. Knowing the location of the online user allows them to flag transactions coming from certain countries which originate a high percentage of credit card fraud [3, 4].

- **Click fraud protection**: The World Federation of Advertisers estimates that click fraud will cause at least $50 billion in losses between 2016 and 2025 [5]. Ad networks use geolocation as one of the features in click fraud detection systems [6].

- **Advertising network targeting**: The location of users is an important factor in personalized advertisements. Determining the location of users correctly can lead to better ad targeting, which in turn can result in higher revenue [7].

- **Law enforcement**: Locating IP addresses can help law enforcement in cybercrime investigations and can aid in detecting unauthorized logins [8–11].

- **Location based content licensing**: Streaming services such as Spotify and Netflix use IP geolocation to enforce geographic content restrictions [12, 13].

- **E-commerce**: Shopping websites can geolocate customers to estimate taxes and shipping charges of products, before customers specify a shipping address [14].

- **Organizations with local points of presence**: Insurance companies, large retail chains, fast food restaurants, and other organizations can use IP geolocation to direct users to their closest facility.

- **Automatic language selection**: Websites available in multiple languages can select the correct one based on the location of the user [15].

Accurate IP geolocation can make the difference between satisfied and dissatisfied customers. 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 [16, 17]. In the case of search engines, incorrect location data can lead to abandoned searches and engine switches. Bennett et al. have shown that training a web ranker to take into consideration user location leads to improvements in 10.4% of test queries [16]. In other cases, such as credit card fraud protection, imprecise location information can have even more severe repercussions such as loss of revenue. To combat fraud, previous work by Akhilomen integrated IP location signals in a credit-card fraud prevention system [4].

Commercial IP geolocation databases are implicitly trusted in academia as a basis for research in various areas, such as core search ranking and user modeling. Bennett et al. [16] personalize search results based on, among other sources, the location of users. However, they derive the actual locations from commercial geolocation services. Zhuang et al. also assume geolocation databases are accurate and use IP location to model click geo-sensitivity [18]. Similarly, White and Buscher resolve user IP addresses to locations to draw conclusions on the knowledge difference between locals and non-locals when choosing restaurants [19]. Finally, White et al. [20] and Yan et al. [21] model cohorts of users by location for personalized search, but again they also use proprietary geolocation services. In this work we find that geolocation databases are unfortunately far from accurate and, therefore, it is worthwhile to improve their performance.

Commercial geolocation services such as MaxMind [22], Neustar IP Intelligence [23], and IP2Location [24] are considered state of the art, although the exact methods they use are proprietary. **Recent work has questioned their accuracy.** Using a ground truth set of 16,586 router IPs, Gharaibeh et al. found a city-level disagreement of 29% across four different vendors using pairwise distances [25]. Shavitt et al. computed the distance between locations reported by commercial databases on identical IP ranges and reported that some pairs of databases have disagreements in the hundreds of kilometers [26]. Poese et al. found errors exceeding 10 kilometers in 80% of the cases, across two commercial databases [27]. Finally, Laki et al. have found that MaxMind places multiple spread out European GÉANT routers in a single location (Cambridge, UK), because that is where the network operator is headquartered and the IP WHOIS records point to that address [28].

1.2 Motivation

The overarching goal of this dissertation is to compile a geolocation database from scratch, by combining IP location information derived from multiple data sources using novel techniques. The reasons for embarking on this effort are manifold:

- IP geolocation is not a solved problem, in either academia or industry. In Chapter 5 we show that the accuracy and IP coverage of previous academic research approaches is typically insufficient for city-level geolocation. The problem also persists in the industry. There are half a dozen companies whose sole focus is IP geolocation. Previous research has shown that currently available commercial databases are inaccurate [25–27]. We also confirm these findings in Chapter 4. Therefore, there is a clear need to continue work in this area and to achieve higher accuracy.

- Data contained in search engine logs has not been previously exploited to improve IP geolocation. Although user location information is heavily used for search engine personalization, the search engine assumes the location of the user is known, often by using commercial IP geolocation databases [16]. It may be possible to use search engine logs to instead derive IP geolocation information.

- Commercial IP geolocation services are proprietary and the methods they employ to derive geolocation database are largely unknown. Consequently, detailing and experimenting with multiple potential methods in an academic setting might shine more light on which approaches work best and why.

- IP geolocation has not been studied at scale. As described in Chapter 5, previous research mainly targets IP coverage on the order of thousands of addresses, as opposed to millions. Furthermore, past work mostly focuses on individual methods. Only a handful of papers *both* cover and combine more than one approach [29–31]. Consequently, combining multiple geolocation methods on a large scale and evaluating their accuracy is worthwhile.

1.3 Research Questions

At the onset of our work we set about to answer several research questions pertaining to IP geolocation.

- How can we obtain a large ground-truth set for IP geolocation?
- What is the impact of inaccurate IP geolocation on user experience?
- How accurate are commercial geolocation databases?
- Can we devise new IP geolocation approaches by exploiting information from the logs of a search engine?
- Is it possible to improve IP geolocation coverage by interpolating (extrapolating) locations across neighboring IP addresses and neighboring IP ranges?
- Can we use public information such as WHOIS databases and traceroutes to augment IP geolocation?
- Which IP geolocation methods provide the most IP coverage and accuracy? What are the advantages and disadvantages of each method?
- How can IP geolocation methods be conflated into a larger geolocation database?

1.4 Outline and Contributions

- In **Chapter 2** we introduce **definitions and concepts** that we use throughout this dissertation.
- **Chapter 3** investigates the impact that **incorrect IP geolocation** can have in the context of search engines. Our contributions include compiling a large ground truth set of 8.4 million IP addresses with known location, and mining the Bing search engine log to compare the metrics of impressions where the location is correct, to the impressions where the location is incorrect. We show that user satisfaction metrics are heavily impacted when user location is unknown.

- In **Chapter 4** we summarize data sources and approaches used by **commercial geolocation services**, we give an overview of previous academic attempts at evaluating the accuracy of commercial databases, and we perform a large scale evaluation of three commercial databases using our ground truth set. We show that geolocation accuracy varies considerably both between databases and within databases, when evaluating accuracy across ten countries. We find that none of the commercial location services can achieve an accuracy above 70% at the city level, and in some cases have an accuracy below 10%.

- **Chapter 5** provides a detailed overview of **previous research in IP geolocation**. We describe prior work across multiple geolocation approaches, including network delay, network topology, reverse DNS, web mining, and others. In Section 5.5 we place our work in the context of prior research, by highlighting the main differences between prior research and the approaches we propose here.

- **Chapter 6** focuses on **privacy and security** in relation to IP geolocation. We first present the current privacy climate surrounding location services. Then, we list the steps we have personally taken to maintain the privacy of users when compiling our ground truth sets, when training our models, and when evaluating our approaches. Finally, we summarize adversarial security work in ways to evade IP geolocation and maintain user privacy.

- In **Chapter 7** we mine the **query logs** of the Bing search engine to improve geolocation. Our contributions include developing an approach to augment commercial IP geolocation databases using information extracted from user queries, and performing a large scale A/B experiment on 1.7 million users and 3.2 million queries to validate our improvements. At the conclusion of the experiment, we observed positive changes in several user metrics. To the best of our knowledge, this is the first time that queries have been used to increase IP geolocation accuracy.

- **Chapter 8** proposes a systematic approach for using **reverse DNS hostnames** to geolocate IP addresses. In our preliminary investigation we determine reverse DNS coverage in the entire IPv4 address space. We also find an upper bound of exact city and airport code matches in hostnames. We then present a machine learning approach for extracting location hints from hostnames. 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. We demonstrate that our approach is both competitive and complementary to commercial geolocation baselines. As a final contribution, we release our reverse DNS geolocation software as open source to help in reproducing our results.

- In **Chapter 9** we present two approaches to extract IP geolocation information from **user clicks** mined from the logs of the Bing search engine. In our preliminary investigation we investigate the geographic focus of URLs to show that clicks on certain links can reveal the location of users. Our first approach uses a density-based clustering algorithm to propagate locations from IPs with known GPS location to IPs with unknown location, through user clicks. Our second approach attempts to eliminate reliance on precise GPS training data by instead using locations mined from the body of clicked web pages. We evaluate the accuracy of our two approaches against two state of the art commercial geolocation databases. Using a large and diverse ground truth set of 70 million IP addresses with known location, we show that our approaches significantly outperform two commercial databases on median error, Root Mean Squared Error (RMSE), and cumulative error distance. Our method also outperforms prior academic work in both accuracy and scale. To the best of our knowledge, this is the first time that user clicks have been used to improve IP geolocation.

- In **Chapter 10** we revisit the task of improving geolocation through **user queries**. We show that by using location clustering applied on queries grouped by IP ranges we can obtain better accuracy results than our preliminary results from Chapter 7. Instead of focusing on improving the accuracy of IP geolocation databases, here we evaluate our revised approach directly against commercial location services.

- **Chapter 11** investigates whether it is possible to improve the coverage of geolocation approaches by using **IP location interpolation**. We study geographic colocation of IP range addresses and we demonstrate that when two IPs are in the same range, they are also often located in the same geographic region. Based on this finding, we propose approaches to extrapolate the location of all addresses in an IP range, starting from the known location of just a few individual IP addresses. Using a large ground truth set, we show that the accuracy of this approach depends on the size of the IP range.

- In **Chapter 12** we propose propagating locations along **traceroute** paths. We show that there is a direct relationship between physical distance and latency differences along the traceroute path. We then combine this property with IP location interpolation to improve geolocation. We evaluate our approach against two state of the art commercial IP geolocation databases, using a large traceroute dataset and a large ground truth set. We show that our approach significantly outperforms the commercial location services. To aid in reproducing our experiments, we also evaluate our approach using a second public ground truth set extracted from PeeringDB, which is a self-reported database of worldwide peering points. Along with this dissertation we are making the traceroute dataset parsing library and the PeeringDB parsing and generation library available as open source. We are also publishing a snapshot of the PeeringDB dataset.

  In **Chapter 13** we extract location information from public **WHOIS** databases provided by the Regional Internet Registries which allocate IP addresses. We describe the technical challenges of parsing and geocoding records that have different schemas. We crawl secondary location information for North American IP ranges, using Referral WHOIS. We demonstrate that geolocation accuracy depends on the granularity of the IP ranges. To aid in future research, we are making both our WHOIS parsing and our RWHOIS crawling libraries available as open source.

  Chapter 14 focuses on conflating location information obtained through different IP geolocation methods to obtain a single geolocation database with higher accuracy and IP coverage. We first evaluate each geolocation approach separately using a single ground truth set of 70 million IP addresses. We then propose 30 features and train a classifier which for a given IP range determines the most likely location candidate, among the ones proposed by our several geolocation approaches. We evaluate the resulting combined database and show that it surpasses two state of the art commercial geolocation services in accuracy.

We conclude with **Chapter 15**, where we revisit and attempt to answer the research questions we posed in the Introduction. We also propose multiple avenues for Future Work.

Chapter 2

Definitions, Acronyms, and Concepts

Table 2.1: Various acronyms used in this dissertation.

Acronym	Explanation
AS	Autonomous System
ASN	Autonomous System Number
BGP	Border Gateway Protocol
ICMP	Internet Control Message Protocol [32]. See *Ping*.
RMSE	Root Mean Squared Error
TTL	Time To Live

Definition 1 (Autonomous System / Autonomous System Number). *An autonomous system or AS is a collection of IP ranges under the control of a single entity that presents a clearly defined routing policy to the Internet [33]. Each such entity is allocated an autonomous system number (ASN) that is used in BGP routing (see BGP).*

Definition 2 (Border Gateway Protocol). *BGP is the standardized protocol designed to exchange routing information among Autonomous Systems on the Internet [34].*

Definition 3 (Error distance). *In the context of IP geolocation, we define* *error distance as the distance between the estimated location of a target IP, and the actual location of the IP, as determined by a ground truth data set.*

Definition 4 (Explicit location queries)**.** *In the context of search engines, we define explicit location queries as queries that contain a location meant to filter results to that geographical area. For instance, the intent of the query "indian restaurant redmond" is to find indian restaurants which are located in the city Redmond [35, 36].*

Definition 5 (Implicit location queries)**.** *In the context of search engines, we define implicit location queries as queries that do not contain an explicit location but still imply local search intent. For example, a user querying for "restaurants" typically expects the results to be limited to their immediate vicinity, even if they did not specifically include a location [35, 36].*

Definition 6 (Local parameter search)**.** *Local parameter search or simply local search is a technique to solve optimization problems which moves through the solution space by applying small incremental changes to model parameters, until a solution deemed optimal is found [37]. Note that here local search is an overloaded term and has a different meaning than in the other definitions. In this context it refers to searching among the possible parameter values of a model.*

Definition 7 (Multilateration)**.** *Multilateration is a technique to geographically locate an object by having multiple stations placed around the target measure and combine the distance between the stations and the target, to estimate its position. Multilateration can also be performed by the target object, without the assitance of stations, by measuring the signals emitted from time synchronized landmarks. GPS is an example of such multilateration where the stations are moving transmitters [38]. In the context of IP geolocation research (see Chapter 5), multilateration is performed by having multiple servers with known location ping a target IP, measure the reply time, and combine the results to estimate the location of the target [39].*

Definition 8 (Ping)**.** *Ping tools are used to determine the reachability and network delay from a source host to a target host on an IP network [40]. ping* *sends ICMP packets [32] to a target address and waits for a reply. Network delay is determined by the time difference between sending a request and receiving a reply.*

Definition 9 (Primate cities)**.** *Primate cities are cities which dominate the surrounding populated places economically and culturally due to their size [41].*

Definition 10 (Reverse Geocoding)**.** *Reverse geocoding is the process of converting a latitude and longitude pair to the corresponding readable address (continent, country, state, county, city, street, etc.). In this paper we reverse geocode locations down to the city level.*

Definition 11 (Root Mean Squared Error)**.** *Root Mean Squared Error or RMSE is an evaluation metric which measures the differences between values predicted by a model and the actual correct values as defined by a ground truth set [42]. To obtain its value, we take the square root of the mean of the squares of the deviations. One difference between RMSE and median is that RMSE easily gets swayed by large outliers, whereas median does not.*

Definition 12 (Time To Live)**.** *Time To Live or TTL is a counter that limits the span of data as it travels a computer network. See Traceroute.*

Definition 13 (Traceroute)**.** *Traceroute tools reveal the path taken by packets to travel from one Internet connected device to another. They also measure the latency to each hop in the path [43]. To determine intermediate routers, an Internet connected device starts by sending out a packet with a time-to-live (TTL) value set to one, followed by more packets where the TTL value of each subsequent packet is always incremented by one. As the packets make their way through the network, intermediary routers decrement the TTL value by one at each step until the value reaches zero. Once the value is zero, the current hop returns a reply to the source. This allows gradually discovering the nodes along the path.*

Chapter 3

Impact of Inaccurate IP Geolocation in Search Engines

We investigate the impact that incorrect IP geolocation can have in the context of search engines. We begin by describing how we compiled a ground truth set of IPs with known location. Next, we mine logs from the Bing search engine to compare the metrics of impressions where the location is correct, to the impressions where the location is incorrect.

3.1 Ground Truth

Limited ground truth information has been a significant limitation in previous IP geolocation research, as discussed in Related Work (Chapter 5). Previous work has typically used tens of IP addresses with known geographical location, mainly located in the U.S. We propose a method to generate a large scale evaluation set by aggregating real time location information from search engine logs. Using our method we have generated a ground truth set of more than 8.4 million IP addresses from across the world.

Mobile devices contain sensors for global positioning systems. Mobile applications can request access to real time location information in order to provide better results. While not all users are comfortable with sharing their location, a representative set of users agree to provide this information. We use this information from search engine logs to generate the ground truth set.

Mobile devices such as smartphones connect both to cell towers and to local Wi-Fi networks. Therefore, they sometimes access the Internet from IP addresses provided by phone operators such as AT&T and T-Mobile, while other times they access the Internet from home or business broadband providers such as Comcast or Charter. Although we collect most of the ground truth data from mobile devices, we perform several filtering steps to ensure that most IP addresses in the ground truth set are from fixed broadband connections. The main reason for retaining only fixed IP addresses is that here we aim to improve existing IP geolocation databases, which assume each IP address has a single location. Since here we are computing a ground truth set, it is reasonable to eliminate IP addresses which appear in multiple cities in the dataset. Second, obtaining the real location of fixed broadband connections is arguably more difficult than obtaining the location of mobile network connections, as most most wired devices such as desktop PCs do not contain sensors capable of determining real time location, while mobile devices do.

First, after aggregating all reported locations per IP address, we restrict the IP addresses to the ones where the location readings stay within a 1.6 kilometer (1 mile) radius. We then assign each IP address a centroid computed by combining these location readings in time. While this constraint does filter out some valid IP addresses, it ensures that the remaining addresses are located in a relatively fixed position. Consider a user who commutes to work in a close-by city. Although the heuristic will likely filter out their mobile IP address as cell phone tower coverage spans large geographical regions, it will retain their fixed broadband IP address, if the user connects the device to both networks. When the user reaches their home and switches to a fixed Wi-Fi connection, their device will still broadcast the real time location to the search engine, thus linking the fixed broadband connection to a fixed location which stays within a small radius. Second, we retain only the IP addresses which appear in the logs on at least three different days, with at least three different location readings. Location information provided by mobile phone operating systems can be stale or inaccurate. This step ensures that we filter out the devices which are only turned on for a short period of time or that broadcast the same stale location.

We also had to overcome other technical challenges. Not all locations in the logs are within the boundaries of cities, and raw locations need to be mapped to individual cities. To address these problems we have retained only the IP addresses which have appeared in the logs on at least three days, which resulted in at least three location readings per IP address, and where at least two of the location readings were distinct. These heuristics are based on our observation that some mobile operating systems continuously report the same last known location information after the user turns off the positioning sensor. Some users in our logs only turn on global positioning sensors temporarily while traveling. The rest of the time the operating system reports the last known (incorrect) location. Finally, we have used the public Bing Reverse Geocoding API [44] to map each location to a city or populated place. This process filters out locations which are outside populated locations, such as lakes, and provides a common naming convention for locations.

We have obtained 8.4 million IP addresses and their corresponding location by mining logs from the Bing search engine for a period of 180 days ending on October 10th, 2014. To the best of our knowledge, this is the largest ground truth set ever used in IP geolocation research. The set spans 220 countries, with a maximum of 2 million IP addresses in the United States. The top 50 countries by IP density have a mean and median size of 163,000 and 45,000 IP addresses, respectively.

To validate that most of the ground truth IP addresses are of fixed broadband networks we have intersected them with information provided by a commercial geolocation database on the types of connections of IP ranges. Results show that 80% of the addresses in our ground truth data are of fixed networks, 2% are of cell phone networks, and the rest unknown or satellite connections. We could have used the connection type breakdown information from Vendor A instead of using our own method, but we have manually found the accuracy of this classification to still be lacking.

3.2 Comparing Sessions With and Without Location Information

We now use the ground truth dataset to analyze the impact of impressions where the location information is incorrect. These cases are nuanced as we expect the effect of incorrect information to become more apparent as the error increases. For instance, for the query *"cinemas"* a user might still click the results even if the search returns businesses from a close-by city.

We extract the impressions issued on the Bing search engine in a seven day period, across all devices. We intersect this set with our ground truth set and then we compute the distance between the ground truth location and the location from the IP geolocation database used by Bing. We compare the set of impressions where the distance between the real location and the assumed location is more than 15 km, to the impressions where the error distance is less than 15 km. We further partition the data based on queries with local intent versus all queries. By a query with local intent we mean queries with local context, such as "plumbers".

Table 3.1 shows the change in metrics for the queries issued from the U.S. market. Both overall and ad click-through rates decrease when the location is incorrect. Prior research has shown that this metric is positively correlated with user satisfaction [45]. However, we observe that Algorithmic click-through rates increases for the "All queries" set. This increase can be attributed to users who avoid clicking on the answers which contain incorrect local content, and instead choose to click on algorithmic search results, which might be more general. For example, if for the query *thai food* the search engine lists restaurants from an incorrect nearby city, the users might instead click on a lower but more generic algo result, such as *restaurants.com*.

The table also indicates that ads click success is negatively affected. Click success occurs when the click results in a dwell time greater than or equal to 30 seconds [45]. This metric has also been correlated with user satisfaction. Therefore, since in our case the metric goes down, user satisfaction might suffer. However, in the case of algorithmic clicks we observe that there is an increase in success when the location is incorrect. This change can be caused by

Table 3.1: Change in metrics between queries where distance error is more than 15 km, versus when the error is less than 15 km. **Note that both result columns show the impact of incorrect locations.** The difference between the columns is that they show two subsets of the data. See text for an explanation of the two positive metrics.

Metrics	**Local Intent**	**All Queries**
Overall Click-Through rate (any link)	-4.3%	-12.8%
Algorithmic Click-Through rate (on search results)	-1.1%	+2.4%
Algorithmic Click Success	-6.1%	+0.2%
Ads Click-Through Rate	-17.9%	-6.1%
Ads Click Success	-15.2%	-7.0%
Ad Revenue	-40.3%	-6.0%

Results statistically significant using a two-sample t-test at 1%.

users that click on general algorithmic results because the local answer shows incorrect results. Finally, we can observe ad revenue decreases dramatically in both cases. Therefore, improving IP geolocation accuracy could lead to higher revenue.

We conclude that when the IP geolocation data is incorrect there is a significant drop in metrics. These results reinforce our hypothesis that incorrect IP geolocation information can lead to lower user satisfaction.

Chapter 4

Commercial IP Geolocation Databases

The most precise way to determine the location of an online service user is to use global positioning systems such as GPS, Galileo, GLONASS and BeiDou [38, 46, 47]. Unfortunately, this information is only available on devices that have location sensors, such as mobile phones. Many other devices such as laptops, desktop PCs, and smart speakers lack the hardware to determine their precise location. Furthermore, even if a smartphone has GPS hardware, users can still opt-out from location sharing. Therefore, we require alternate ways to locate online devices.

Other location detection methods include Wi-Fi positioning systems, as well as crowdsourced locations. In the former case, the location of the user is determined by comparing the list of wireless access points visible from the device to a list of access points with known locations. These Wi-Fi location databases suffer from lack of coverage, especially in rural areas [48]. In the latter case, having users self-report their location might be accurate in the short term, but such information can become stale in the long run if users move without updating their location.

IP geolocation databases are an alternative with higher coverage. They consist of IP ranges mapped to location information at the city level. In some cases they can also contain further information on IP ranges, such as details about the ISP which controls the range, or the speed and type of

Internet connection offered in the range. They typically have high IP coverage and medium to high accuracy. These databases are provided by commercial vendors such as:

- **Akamai EdgeScape** [49] is an IP geolocation database that was introduced in June 2000. Akamai was founded in 1998 and is located in Cambridge, MA, USA.
- **Neustar IP Intelligence**, formerly known as Quova, is a geolocation service introduced in 2000. Neustar [23], founded in 1999 in Mountain View, CA, USA bought Quova in 2010.
- **IP2Location** [24], founded in 2001 and located in Bradenton, FL, USA
- **IPligence** [50], founded in 2006 and located in Barcelona, Spain
- **MaxMind** [22], founded in 2002 and located in Waltham, Massachusetts, USA
- **Digital Element's NetAcuity** [51], founded in 1999, in Norcross, GA, USA

4.1 Data Sources

The data sources and approaches used to compile commercial geolocation databases are closely guarded secrets. The proprietary nature of these databases makes them difficult to analyze. In some cases, such as NetAcuity, Neustar, and EdgeScape, it is difficult to access the databases without signing a contract. In the majority of cases, the terms of usage of the geolocation databases disallow comparative benchmarking. Yet, they are widely used and trusted by both industry and academia.

Digital Element was the first company to provide geolocation services and is the self-professed leader in commercial IP geolocation. Descriptions on their website state that they combine multiple data sources, including location data obtained directly from Internet Service Providers and location data derived from mobile phone traffic, although they do not clearly state the source of this data [51–53]. Digital Element applied for and was issued multiple patents on IP geolocation [54–57]. Our analysis of these patents suggests they may be collecting data through network latency and topology approaches (ping, traceroute), may be extracting locations from the WHOIS databases maintained by

Regional Internet Registries, may be parsing reverse DNS hostnames for location hints, and may be using data collected through crowdsourcing. Rob Friedman, who is the co-founder of Digital Element, also lists these same data sources as an answer to an IP geolocation question on Quora, but cautions that data from WHOIS databases is generally inaccurate [58]. However, it is unclear if or how they use these resources in production.

MaxMind, IP2Location, and Neustar IP Intelligence may also be using WHOIS data as part of their databases. Lee et al. find evidence that MaxMind is using data derived from the APNIC WHOIS Internet registry database in Korea [59]. They find typos in city names returned by MaxMind that are identical to typos found in WHOIS database entries. They also find that MaxMind sometimes incorrectly parses Korean addresses stored in these entries, due to a lack of understanding of how Korean mail addresses work. As a result, MaxMind occasionally mistakenly picks small towns with names similar to that of larger cities. IP2Location compiles a yearly study of IP allocation that uses WHOIS data [60]. Neustar states on their website that they use Internet registry data [61]. They also suggest they may be using network delay and topology information.

Using data from a large European ISP, Poese et al. show that MaxMind and IP2Location partially rely on the official IP allocations and BGP network advertisements [27]. The authors find significant overlap between the granularity of advertised networks and that of the IP ranges in the commercial databases.

Liu et al. [62] reveal a rare glimpse in the data sources used in a second tier commercial geolocation database compiled by Tencent. The authors state that this database is compiled by combining WHOIS data, locations from reverse DNS hostnames, and crowdsourced IP location information.

Lastly, Chandrasekaran et al. state that commercial databases rely on a combination of domain registry information, ISP provided data, host name hints, latency measurements, and other heuristics [31]. Furthermore, they also found evidence of manually curated information used in major geolocation databases.

In summary, these databases are compiled by combining data from multiple sources:

- **WHOIS databases**. The primary data sources for these databases seem to be the databases maintained by the five regional Internet registries, which manage how IP ranges and network autonomous system numbers are assigned to organizations around the world [63]. These registries maintain WHOIS databases with mappings of IP ranges to the geographical location of the organizations that use these IPs. However, there are cases where large ISPs or multinational companies are assigned large IP ranges. Since these organizations deploy infrastructure over large areas, the geographic information contained in the registry databases is insufficient to obtain accurate locations from any IP address in the range. In Chapter 5 we summarize related work that uses WHOIS databases and in Chapter 13 we evaluate using this data ourselves for IP geolocation.

- **Network delay** information collected by issuing through ICMP packets with tools such as ping. We describe past research in this area in Section 5.1.

- **Network topology** data, including traceroute paths, BGP routing table snapshot, and Autonomous System network information, as we describe further in Section 5.2 and Chapter 12.

- **Reverse DNS hostnames** that contain location hints. We summarize related work in Section 5.3 and and we present our own novel approach for geolocation using these hostnames in Chapter 8.

- **Crowdsourcing** or manually curated IP location information, as partially described in Section 5.4. For example, weather websites may ask users to self-report their location to obtain a forecast for their city. This location information can then be stored along with the IP address of each user.

- **Mobile data traffic** which contains IP addresses, along with precise GPS locations. This data is usually collected through mobile apps. See Section 3.1.

- **Partnerships with ISPs**. Commercial geolocation databases can partner with large ISPs, which maintain internal records of how they geographically subdivide large IP ranges provided by the Internet registries.

4.2 Accuracy of Geolocation Databases

We summarize self-reported accuracy numbers provided by commercial databases, followed by an overview of independent evaluations performed in academia. We then carry out our own large-scale evaluation of three commercial databases, using a ground truth set of 8.4 million worldwide IP addresses. All the results presented in this section refer to IPv4 addresses.

MaxMind and IP2Location are among the few commercial geolocation databases that reveal their city-level accuracy. The results vary significantly on a country by country basis. Maxmind reports an accuracy smaller than 50% for 36 out of 100 countries when using an error distance threshold of 10 kilometers [64]. For some well-known countries such as Italy, Venezuela, Philippines, and Japan, this accuracy is below 35%. Their mean unweighted accuracy across 100 countries is 58% at error smaller than 10 kilometers, 70% at error smaller than 25 kilometers, and 76.5% at error smaller than 50 kilometers. IP2Location also reports a similar accuracy result of 77% for error smaller than 50 kilometers [65]. In the United States, using a threshold of 50 kilometers, MaxMind reports an accuracy of 86%, while IP2Location states an accuracy of 99.53%. However, all of these numbers are self-reported and there is no information on the freshness of these results, or the size of the ground truth set.

Using the internal naming scheme of routers obtained from a large European ISP, Poese et al. show that MaxMind has an accuracy of less than 20% within an error distance of 10 kilometers, while IP2Location fares even worse with errors ranging from 200 to 800 kilometers [27].

To evaluate multiple commercial geolocation databases, Shavitt et al. [26] employ a ground truth set of 25,000 worldwide IPs provided by CAIDA [66]. They find that NetAcuity, MaxMind, IP2Location, and IPligence have an accuracy of 79.1%, 29.4%, 14.16%, 0.9%, respectively for error distance of 100 kilometers or less.

Kester evaluates commercial databases using 3,206 IPs consisting of CAIDA Ark [67] and RIPE [68] servers. For the same threshold of 100 kilometers, he finds that MaxMind and IP2Location have an accuracy of 63% and 67%, respectively.

Table 4.1: Basic statistics on the the IP geolocation databases used in this work.

Provider	**IP Coverage**	**Countries**	**Cities**
Vendor **A**	3.48B (˜94.1%)	249 (100%)	87,842
Vendor **B**	3.48B (˜94.1%)	249 (100%)	66,480
Vendor **C**	3.6B (˜97.3%)	247 (99.2%)	102,837

These results show that there can be quite a difference in accuracy between commercial geolocation databases, depending on the database and the ground truth used to evaluate them.

To get a better idea of accuracy, we use the ground truth dataset of 8.4 million IP addresses described in Section 3.1 to evaluate three well-known commercial IP geolocation databases. These databases represent the state of the art in the industry and, as described in Chapter 5, have higher coverage and better accuracy than most previously published research. Due to legal reasons we will use the generic names Vendor *A*, Vendor *B*, and Vendor *C* to denote them. The providers were chosen from among the companies listed above. Nevertheless, the findings are still valuable even without specifying their exact names.

Table 4.1 presents basic statistics about these databases, generated on data available on October 10th, 2014. The IPv4 address space is 4.29 billion IP addresses, which includes 592 million addresses yet to be distributed to organizations, or reserved for other uses. Vendor *C* has the highest IP coverage at 3.6 billion, followed by the other two vendors each at 3.48 billion. The databases report countries using the ISO 3166-1 standard, which yields a maximum of 249 possible countries. Vendors *A* and *B* cover all possible countries, while Vendor *C* covers 247 of them. The last column in the table shows city level coverage. The difference between the vendors with highest and lowest city coverage is 36,357 cities. The databases have very high coverage at the IP and country level, but varying degrees of coverage at city level.

In Chapter 3 we have shown that missing IP geolocation information at the city level has a negative impact on search engine user metrics. The city related fields are empty for 6.9%, 6.9% and, 37.4% of the covered IPs in the three databases, respectively. The largest difference in coverage between the

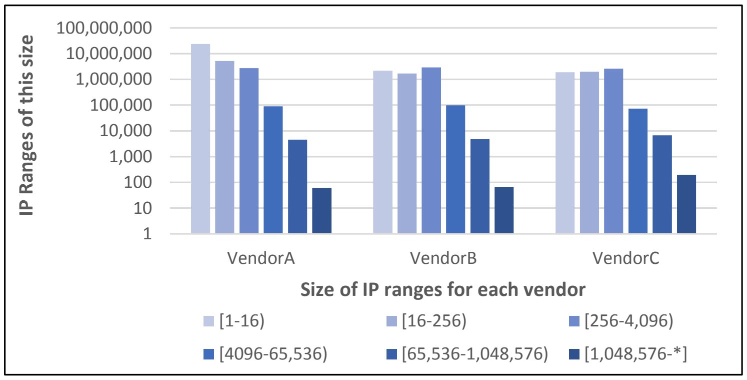


Figure 4.1: Distribution of IP range sizes after merging contiguous ranges that have identical locations.

first two vendors and the last one was in the United States, with 50.8, 50.7, and 518.6 million IP addresses without city information, respectively. Although the commercial databases cover a large number of distinct cities, computing the city coverage at the IP granularity shows that there is room for improvement.

The maximum IP range size in all tested databases is 256. In other words, the size of a group of IP addresses with the same location is at most 256 IPs. However, we have noticed in the raw data that several IP ranges are both contiguous and have the same location. For illustrative purposes we have merged contiguous IP ranges which contained identical information. For instance, if two IP ranges have sizes 16 and 256 respectively, are contiguous, and have identical location information, we group them in a single range of size 272. Figure 4.1 shows the IP range size distribution after merging contiguous ranges with identical information. To make the figure easier to plot we group the sizes of IP ranges in buckets. The X axis shows the buckets, which have limits given by powers of two. For example, the bucket [1-16) contains all IP ranges of size 1 to 15. The Y axis shows how many IP ranges are in each bucket, using a logarithmic scale. We note two findings. First, there are cases where IP ranges are more than 1 million IPs in size. It is unlikely that all of these IP addresses have the same location information in reality. Second, the database from Vendor *A* contains ranges which are more granular than the other two vendors. It contains 23.8 million IP ranges where the size is less than 16 IPs, compared to 2.2 million for Vendor *B* and 1.9 million for Vendor *C*.

We now turn to the ground truth data to evaluate the accuracy and error distance of the databases. Since the data is provided by different companies, it is possible that the names of locations are not consistent. To solve this problem we have used the public Bing reverse geocoding API [69], which maps coordinates to street addresses. The reverse geocoder is aware of the boundaries of cities. This reverse geocoder is the same one that we have used while generating the ground truth dataset in Section 3.1. Since we are interested in geolocation at the city level, we discard the street addresses. We normalize the locations in each database by assigning a unique identifier to each distinct city. Using the same reverse geocoding method on the databases and on the ground truth data ensures a level playing field in terms of location names and identifiers. One important side effect to note is that all ground truth locations within the limits of a city are converted to the coordinates of the center of that city.

We compute the accuracy of exact city matches by counting how often the location of ground truth IPs matches the location given by the databases. In Figure 4.2 we show the accuracy for the three databases across the top ten countries by ground truth IP density. There are two interesting findings. First, none of the databases achieve an accuracy above 70% at the city level, and in some cases have an accuracy below 10%. Second, Vendor C outperforms the other two providers, often significantly, across all ten countries. The same findings hold for all countries. This finding is particularly interesting as in 37.4% of the database from Vendor C there is no city level information. This suggests that Vendor C has a high accuracy but low city-level IP coverage, while the converse is true for the other two vendors. We have also found that country-level accuracy is significantly better than city level accuracy, with a median of 95.3%, 96.7%, and 89.2% in the top 50 countries for the three vendors, respectively.

Exact city level matches might not present a complete picture of the performance of IP geolocation databases. Therefore, we also compute the error distance between the cities from the ground truth IPs, and the cities returned by the geolocation databases for these IPs. The error distance is given by the distance between the centers of the cities. For a given ground truth IP, if the ground truth and IP geolocation cities are the same, the distance is zero.

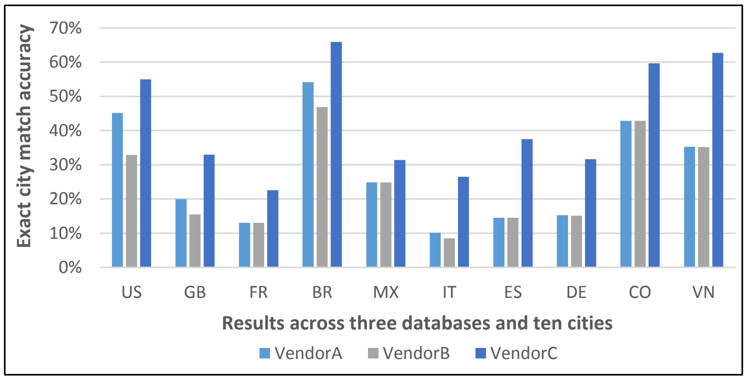


Figure 4.2: Accuracy at the city level for the three IP geolocation databases, across ten countries. Data points show exact city match accuracy.

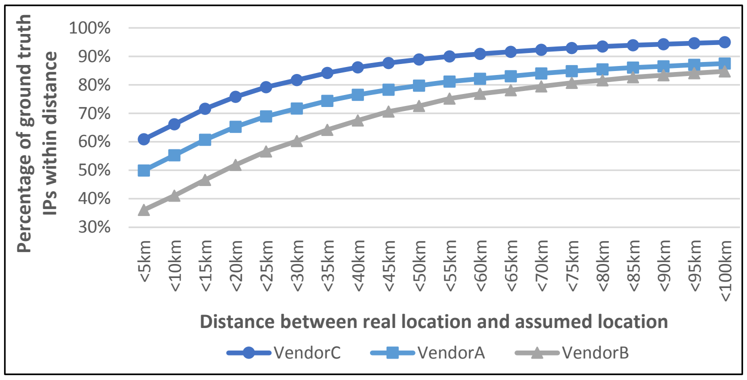


Figure 4.3: Cumulative error distance for the three IP geolocation databases **in the United States**. Data points show percentage of ground truth IPs where the real location and the IP geolocation assumed location fall within a certain distance. The distance interval on the X-axis is five kilometers.

Figure 4.3 shows the cumulative error distance for the three databases, in the United States. The shape of the curves is similar in the other countries. The X axis shows the error distance at an interval of five kilometers and the Y axis shows the percentage of ground truth IPs with that error distance. For example, consider the three data points above the X-axis label titled *<10km*. The numbers show that the real location of 66.2%, 55.3%, and 41.1% of ground truth IPs is within 10 kilometers of the location provided by Vendors *C*, *A*, and *B*, respectively. The closer the curve is to the upper left corner, the smaller the error distance, and the better the results.

We have also performed a comparison of the accuracy and error distance results between our ground truth method, and the self-reported data provided by Vendor C for the United States. We have found that the exact city accuracy difference is within two percentage points, while the error distance at 10 kilometers is within seven percentage points. For the exact city match accuracy their results are within 10 percentage points of the results we have obtained in Figure 4.2.

In conclusion, commercial geolocation databases can have low accuracy at the city level. Therefore, further research in IP geolocation is worthwhile.

Chapter 5

Related Work

In this chapter we summarize prior work in IP geolocation, which has been an active research area for more than two decades. Work in this area encompasses a diverse array of IP geolocation approaches. In Section 5.1 we begin by discussing techniques that use network delay and multilateration to perform geolocation. Section 5.2 further introduces information derived from network topology, such as traceroutes, BGP routing information, and Autonomous System network data. In Section 5.3 we summarize work in extracting location hints from reverse DNS hostnames. Section 5.4 presents an overview of extracting IP location information through web mining, by crawling web pages, propagating locations in social graphs, mining social checkins, consulting online WHOIS databases, and crowd-sourcing location information from Internet users. Lastly, in Section 5.5 we put our work in the context of prior research, by presenting a comparison of past approaches and their results, by discussing their trade-offs and limitations, and by highlighting the main differences between prior research and the approaches we propose here.

5.1 Network Delay

The line of research on network delay relies on the observation that the latency experienced by network packets as they travel between two Internet hosts is proportional to their geographic distance. Early work in IP geolocation by Padmanabhan and Subramanian discusses **GeoPing** [70, 71], which uses delay measurements made from geographically distributed locations to infer the coordinates of the target IP. GeoPing estimates the location of the target IP by picking the location of the probe server with lowest latency to that target IP. This method has a median error of 382 kilometers when locating 265 IPs located at universities across the United States. Ziviani et al. take a very similar approach and achieve a median error of 314 km using 397 IPs also mainly located on university campuses [72].

CBG [39, 73] goes further by drawing circles on the surface of the Earth around each probe server with known location, where the radius of each circle is given by the network delay value. It then picks the center of the intersection of these circles as the likely location of the target IP. Figure 5.1 shows an example of the area created by the intersection, for a specific target IP. This technique is called *multilateration*. To estimate the conversion between network delay and physical distance, CBG builds a simple model for each probe server by measuring the delay to all the other servers. Since the locations of all probe servers is known, a *bestline* that fits below all measurements can then be determined. Figure 5.2 shows an example of such a ***bestline***, compared to a ***baseline***, which is the expected theoretical line when using 2/3 the speed of light as the maximum propagation distance in fiber. Experimental results on 95 IP addresses in the United States and 42 addresses in Western Europe produce a median error distance of below 95 kilometers for the first dataset and 22 kilometers for the second dataset, respectively.

Later work based on CBG focuses on ways to better determine the bestline. Youn et al. [74] propose a statistical method called Statistical Geolocation, or **SG** for short, to better estimate the bestline by applying kernel density estimation to delay measurements. On a dataset of 85 IP addresses, SG achieves a median error of 53 kilometers. **Spotter** [28], a technique proposed by Laki et al., further refines the task of fitting a baseline. Instead of generating a separate bestline for each server, Spotter derives a single bestline for all of them, by combining readings from all active probes together into a common delay-distance model. Experiments on ≈100 PlanetLab [75] servers and 23,000 Cogent [76] servers yield median errors of 75 and 30 kilometers, respectively. Dong et al. propose **SDP**, which clusters the readings on the delay-to-distance graph by using k-means clustering. For a given target IP, it

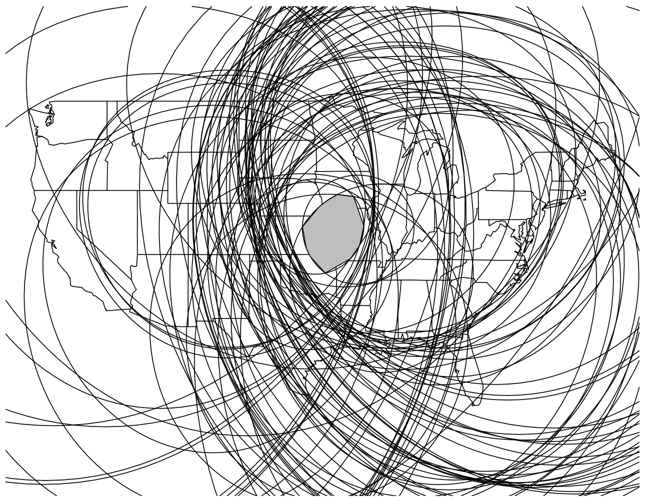


Figure 5.1: Location estimation of a target host using multilateration. Figure from Gueye et al. [73], reused with permission from ACM.

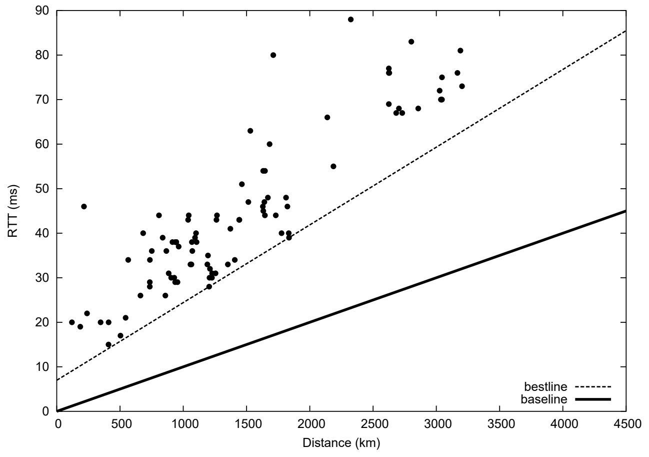


Figure 5.2: Sample scatter plot of geographic distance and network delay. The ***best**line* is below all plotted data points, while the ***base**line* is the theoretical line given by 2/3 the speed of light. Figure from Gueye et al. [73], reused with permission from ACM.

first picks the closest cluster and then it performs more extensive probes from the subset of probing nodes local to that cluster [77]. This approach achieves a median error of 27 kilometers on 81 PlanetLab servers in North American, and a median error of 42 kilometers on 90 PlanetLab servers in Europe.

Although incremental, more recent work by by Khan et al. extends CBG by introducing a similar two-step process. In the first step they identify a coarse region using worldwide probe servers, and then in the second step they further refine the location using regional servers. Their technique named **AIG** [78], which is short for Adaptive IP Geolocation, achieves a median error of 22 kilometers using 89 IPs in North America, and a median error of ≈7 kilometers using 30 IPs in Pakistan. Jiang et al. also propose a very similar two tiered approach we call **Neural-RBF** [79], since it employs a radial basis function (RBF) neural network. In both tiers, the training data is ping measurements issued from 14 RIPE Atlas probes [68] that target US landmarks with known location. After training, the run-time input consists of latency readings sent from RIPE Atlas probes to a previously unseen US target IP, and the the output is estimated location coordinates. The difference between the two tiers is that the first tier is trained to determine a coarse location (48 lower US states) using all training data, while the second tier instances are trained on data specific to a specific region. Using 1,547 IPs, they obtain a median error of just 4.1 kilometers. However, the entire ground truth set is skewed towards only IPs that are reachable through ping, and that are mostly on high-speed business networks such as academic institutions and local governments.

Ciavarrini et al. derive the Cramér–Rao lower bound (**CRLB**) for IP geolocation [80], a theoretical model that can be applied to pure network delay approaches such as CBG to determine the best placement of measurement servers and to measure the impact that the number of servers has on accuracy. It can also be used to ascertain the effect of varying the distance between the servers and the targets, and to observe the influence that changing the number of ICMP packets sent to a target has on determining geographic distance. CRLB has previously been used for evaluating the optimal theoretical positioning of probes in other localization scenarios, such geolocation in wireless networks, or tracking acoustic sources [81–83]. Ciavarrini et al. are the first ones to use it for building a theoretical model of network latency IP geolocation. They derive the parameters in their model from real data collected through 39 PingER [84] machines. The model simulates a theoretical area of 3,000 by 3,000 square kilometers, which is about the size of USA and twice the size of Europe. They arrive at several interesting findings. First, they demonstrate that using a very large number of active probe servers - on the order of 500 or above - provides only a marginal gain in terms of accuracy. Doubling the number of servers from 500 to 1,000 causes the Root Mean Squared

Error (*RMSE*) to improve only from ≈22 km to ≈15 km. Even when using hundreds of servers the error can still be in the order of tens of kilometers. Second, they show that as the distance between the probes and the targets increases the RMSE value increases as well, which means the error gets worse. Third, increasing the number of ICMP samples sent to a target IP to find the smallest latency value has diminishing returns. In fact, increasing the number of packets from 10 to 50 only improves RMSE by ≈3%. Fourth, the authors demonstrate that obtaining an error below 20 kilometers requires so many probing servers as to be unpractical. We further discuss this last finding in Section 5.5.

5.2 Network Topology

Network Topology geolocation methods also use knowledge of the network structure to achieve increased accuracy. **GeoCluster** [70, 71], also proposed by Padmanabhan and Subramanian, combines BGP routing information with sparse IPs of known location to assign geographic locations to whole address prefixes. In this dissertation we call this process *IP location interpolation.* On a set of 265 target IPs located at universities in the United States, GeoCluster achieves a median error of 28 kilometers. This approach performs well with hosts located on university campuses, but it performs much worse on a larger and more realistic dataset of 181,246 IPs, with median error degrading to 685 kilometers.

Jayant and Katz-Bassett extend CBG's ping-based approach by adding information from traceroutes [85]. They hypothesize that targets that follow similar traceroute paths also have similar delay-to-distance conversion characteristics. They propose Path-Based Estimation (**PBE**) and Router-Based Estimation (**RBE**). The former selects a subset of the monitoring servers when determining the delay-to-distance bestline. To obtain the subset, the server traceroutes to IPs in the training set, using a TTL set to x, a progressively increasing number of hops. For each x it finds the longest shared prefix between all the traceroutes with x hops, then builds a delay-to-distance bestline only for that subset. The latter approach is a simplified version which instead of measuring for all values of x, it only measures for a given x. Using a ground truth of 96 IP addresses, PBE achieves a mean error distance between 1000 and 2,400 kilometers, while RBE achieves a mean error distance between 150 and 2,400 kilometers, depending on the value of x. The median errors for the two methods are 376 kilometers and 346 kilometers, respectively.

Katz-Bassett et al. later also propose **TBG**, which stands for Topology-Based Geolocation [86], again largely based on CBG. The authors treat geolocation as an optimization problem on a graph with n active probing servers with known location, and m targets with unknown location, where the edges are given by traceroute paths with network delay. TBG achieves higher accuracy than CBG on a dataset consisting of 128 IPs located at universities in the United States. The authors evaluate three variants of TBG, the best of which also uses reverse DNS location hints based on the hostname of IP addresses (see Section 5.3). This variant, named *TBG-undns*, achieves a median error of 67 kilometers.

Although mainly inspired by CBG and TBG, the **Octant** framework [29] also makes use of locations from WHOIS servers and from reverse DNS hostnames, whenever available. They search for zip codes in the WHOIS records returned when querying for target IPs. The authors also use manual hostname parsing rules provided by the Rocketfuel project at University of Washington to parse reverse DNS hosthames [87]. Octant introduces multiple novel techniques to improve constraint-based multilateration, including using negative information on where a node *cannot* be located, and using the location information of intermediary routers on the traceroute path from the source to the target IP. Whereas CBG draws *disks* around landmark servers, Octant uses *rings*, which introduces negative constraints. The negative constraints are given by the hollow part of the ring, which signifies the area that *cannot* contain the location of the IP. For a given target IP, the output of the framework is a surface bounded by Bézier curves.

The disadvantages of the Octant framework are that the ground truth as well as the WHOIS and reverse DNS extraction rules are US centric. Furthermore, the ground truth is limited to only 104 IP addresses at academic institutions in the United States, which typically have good Internet connectivity. Nevertheless, the results are promising, with a median error of 35 to 40 kilometers (22 to 25 miles). This framework demonstrates that combining multiple data sources can have a positive impact on the results. The authors mention that the reverse DNS data helped accuracy for half of the ground truth IPs. While evaluating Spotter [28], Laki et al. try to reproduce Octant results but achieve much worse results. They find a median error of 125 kilometers using ≈100 PlanetLab [75] servers, and a median error of 120 kilometers using 23,000 Cogent [76] servers.

Alidade [31] is a successor to both CBG and Octant. Chandrasekaran et al. state that, like Octant, Alidade is at its core a CBG-based approach. The authors initially reimplemented Octant, then they added more techniques to obtain Alidade. This system is perhaps the most comprehensive so far, as it combines ICMP and TCP latency measurements, traceroutes, reverse DNS, WHOIS, IP location interpolation, and AS network information. Alidade performs multiple (typically three) iterations over the data. In each iteration, it derives location constraints from observations (similar to CBG multilateration), solves the constraints, then combines the solution with non-measurement data such as reverse DNS. Active latency measurements always take precedence and override non-measurement data such as reverse DNS information. Whereas CBG expresses constraints as disks around landmark servers, Alidade uses N-sided polygons, where N is usually set to 32.

Alidade is a combination of multiple components. The *extrapolator* tries to infer the location of target IPs by looking at the reverse DNS hostnames on the traceroute path to the target. It extracts location hints from hostnames using a proprietary hostname parser called *HostParser*. The *preloader* also uses HostParser to locate individual target IP hostnames at the city-level. The *aggregator* uses IP location interpolation to increase IP coverage. They also combine WHOIS data with Autonomous System information to assign higher location confidence to IP ranges stemming from lower tier and stub networks.

An interesting novel feature of Alidade is that instead of returning single coordinates for a target IP, they attempt to output the polygon shape of the city. They obtain these shapes from public sources such as TIGER/Line [88] and GADM [89].

Chandrasekaran et al. evaluate Alidade on six ground truth sets. All but one of the datasets is very small, on the order of tens or hundreds of IPs.

The *EuroGT* dataset is the largest one at 100,000 IPs, but it is sourced from a single European Tier-1 ISP. Also, the dataset exhibits low geographic diversity, by covering only 73 European cities. The evaluation shows mixed results, with Alidade significantly outperforming commercial databases on the EuroGT dataset with a median error of ≈10 kilometers, while at the same time underperforming some these databases on the smaller datasets (*PlanetLab* [75], *Measurement Lab* [90], *Ark* [67], *GPS*, *NTP*). The text suggests Alidade outperforms the commercial databases on the Measurement Lab dataset, but the results curve shows otherwise. One commercial baseline has better error distance when error is less than 2 kilometers, and another one has better error distance between 14 and 200 kilometers. Chandrasekaran et al. state that the reason why Alidade outperforms the best commercial baseline is that its worst error is 370 kilometers, compared to the baseline which can have data points with higher *maximum* error. However, this is only true for the outliers, and it does not say much about the overall accuracy of the database. Although Alidade is one of the most comprehensive approaches described in geolocation literature, its results on the more geographically diverse datasets are not strong. Furthermore, the system heavily relies on HostParser, which is a proprietary reverse DNS parsing library with unknown methodology and accuracy.

The **Geo-RhOL** [91] approach proposed by Laki et. al builds upon CBG and TBG by estimating the network delay of traceroute path segments at a more granular level. They divide network delay into *routing delay*, which is the time spent by the packet inside a router; *transmission delay*, which is the time needed to place a packet onto a link; and *propagation delay*, which represents the time needed for the packet to travel along cables. They also further divide routing delay into *processing delay*, *queueing delay*, and *other delays*. They estimate the values of these intermediary delays, in order to determine the overall delay more accurately. They then perform experiments on a small number of IP addresses from academic institutions in the United States and Europe. Their five proposed variants, some using only network delay and some also adding network topology information, achieve mean errors between 149 and 437 kilometers.

GeoBuD by Gueye et al. [92] is a simpler attempt to take into consideration routing delays when compared to Geo-RhOL, as its model is more coarse and only considers delays stemming from buffering. The authors extend CBG to account for the delays in each segment of the traceroute path. They replace the linear model used by the original CBG approach with one which decomposes the delays on a per-hop basis. Gueye et al. then estimate the delay for each hop by using a separate dataset of intermediary routers with known locations. For a pair of routers with known location, the buffering latency is given by the difference between the actual observed latency and the ideal theoretical latency. This approach achieves a median error of 100 kilometers on 57 European IPs and 144 kilometers on 87 US IPs.

To go beyond simple latency measurements, Eriksson et al. use traceroute hop counts between the probe and the target, and also introduce a population density estimate. They implement a Naive-Bayes classifier (**NB-LHP**) which combines this information to assign the location of each target IP to a county in the United States [93]. For ground truth they use 16,874 router IPs that were mapped to a US county by using MaxMind [22], which is a state of the art commercial geolocation database. They obtain a mean error distance of 408 kilometers. In addition to the large distance error, the other obvious disadvantage of this evaluation approach is that it uses proprietary databases as ground truth.

Later, Eriksson et al. further refine the usage of hop counts for geolocation in **Posit** [94, 95]. For each target IP and known landmark IP, they create a vector with hop counts from their server monitors to these IPs. For a target IP, they then pick the location of the landmark IP that has the smallest variance between the hop count vectors for the target and the landmark. Using 283 target IPs in the United States, they achieve a median error of 314 kilometers (195.16 miles). They also test a version of *Posit* where the vectors contain latency results instead of hop counts and they only consider the vector elements that fall below a latency threshold. Using the same target IPs, they achieve a significant reduction in median error, down to 42 kilometers (26.15 miles).

5.3 Reverse DNS

Reverse DNS geolocation research focuses on parsing location hints from the reverse DNS hostnames assigned to IP addresses. For example, the reverse DNS hostname of IP address 50.106.32.158 is `static-50-106-32-158.both-.wa.frontiernet.net`. Based on the location hints *both* and *wa*, we can conclude that the IP address may be located in *Bothell, Washington.* Using locations hints in naming conventions is a widespread practice among Internet Service Providers. A survey sent by Chabarek and Barford to 22 North American ISPs has revealed that 20 of them use geographic encodings in their hostnames [96].

GeoTrack, another early attempt by Padmanabhan and Subramanian, tries to infer the location of IP addresses from the Reverse DNS names of the routers along the traceroute path [70, 71]. For a target IP, they perform traceroutes from 14 probe locations. They then search for location hints in the nodes along each traceroute path. They assign the location of the closest parsable hostname on the path, to the target IP. They use airport codes from United States and city/country codes from United States, Canada, and Europe, as location hints. This approach yields a median error distance of 590 km on a test dataset of 2,830 IPs. There are two main disadvantages with this approach. First, it requires traceroutes from multiple vantage points to each target IP. Second, the location hints and the rules on how to locate them in hostnames were crafted manually through trial and error for each Internet Service Provider in the test set. Therefore this approach is not sustainable for a large number of IP addresses and is not scalable to the entire world.

Undns is the most well-known and widely used reverse DNS geolocation approach [87]. It consists of manually curated regular expressions that are grouped by ISP domain names. These rules are used to extract location hints from reverse DNS hostnames. For example, the rule `([A-Z]{3,4})[0-9]?-.verizon-gn-.net` matches subdomains with 3 or 4 uppercase letters, followed by an optional numeric digit. This rule matches hostnames such as `PHIL.verizon-gni.net`. A domain specific location dictionary is then used to match the extracted slot `PHIL` to *Philadelphia, PA*. This approach is similar to the one used by GeoTrack. The obvious disadvantage of this approach is that each domain requires manually generated and potentially error prone rules. Also, its accuracy is unknown because the paper does not present a specific IP geolocation evaluation. Several geolocation and network topology papers use undns as-is to draw conclusions or perform experiments [27, 97–99]. We demonstrate in Chapter 8 that undns can have low accuracy.

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 [100]. 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 cannot scale to the entire IP space, and its method of splitting hostnames is rudimentary. Also, the approach is only evaluated against router hostnames, when most geolocation applications need to geolocate end user residential IP addresses. Furthermore, the approach was only tested on ISP domains that have reverse DNS hostnames with location hints.

DDec [101] combines undns and DRoP rules by giving precedence to undns and using DRoP as fallback. Unfortunately, we demonstrate in Chapter 8 that DDec and its constituent parts - undns and DRoP - perform poorly on worldwide ISP domains due to incorrect rules and catch-all rules.

HLOC, which is more recent work by Scheitle et al. [102], 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.

5.4 Web Mining and Other Approaches

Web mining approaches use diverse information extracted from the web. Although more rare, these approaches often achieve better accuracy and IP coverage results.

Structon [30] is an approach proposed by Guo et al. that mines the contents of Chinese websites for mentions of locations, using regular expressions. The authors assign these locations to the IP addresses of the web servers hosting this content. They then use IP location interpolation to increase both accuracy and coverage by estimating the location of entire IP ranges from the location of few individual constituent IP addresses. They assume all IP addresses in the same /24 segment are in the same city and they combine multiple types of IP location interpolation. First, if a majority of IPs in a range are in the same city, they assign that city to the entire range. Second, they continue iteratively applying this heuristic on increasing IP range sizes until they reach a netmask of size /18 (16,384 IPs). If smaller IP ranges inside a larger IP range agree on location, they assign the location to the larger IP range as well. Third, they use a BGP routing table snapshot combined with Autonomous system network information to assign locations to all ranges of small ISPs, if the location of one of the ranges is known. Finally, they perform traceroutes to IPs in /24 segments that still do not have a location. They retain only traceroutes where all nodes in the path responded to ICMP packets. For a target IP, they assign its location to be that of the closest router with known location on the traceroute path. They also propagate locations backwards, starting from a range with known location, assigning it to a router preceding it on a traceroute path, then assigning the interpolated location of the router to all its neighboring ranges. All these approaches taken together achieve an accuracy of 87 percent at the city level. Instead of computing error distance as in other previous work, they map coordinates to cities, and they check if the city of their location candidate matches exactly to that of the ground truth data point.

While these results are impressive, this work has several problems. The starting assumption that the web server hosting a website is in the same location as the organization that owns the website and its users, may not hold today. With the advent of cloud computing, many websites are now hosted in centralized data centers, and not in decentralized local business offices. Second, the evaluation is performed on a crowd-sourced ground truth set with unknown freshness and accuracy. Third, the manually created extraction rules used to mine location information are tailored specifically for China and the authors admit they may not work in the rest of the world. Fourth, the paper states the task is made easier by the fact that China only has a few hundred cities, compared to 35,000 cities and towns in the United States. This difference can skew the results favorably when evaluating this approach on Chinese data at city level granularity. Nevertheless, several approaches described in this work are interesting, especially for IP location interpolation.

Wang et al. combine the CBG approach [39, 73] with location information extracted from web pages. We name their proposal **WebCBG** [103]. They consult online "Yellow Pages" directories to find websites of businesses and their corresponding postal addresses. After several filtering steps they retain a subset of the IPs of these web pages, along with their corresponding locations, and use them as landmarks. Given a target IP, they then home in on its correct location by first applying CBG to determine a coarse location and a list of nearby landmarks, then they perform multiple traceroutes to both the target IP and the business IP landmarks, from multiple vantage points. Using the traceroutes and the closest common path between the target IP and a business landmark, they estimate the location between the IP and each landmark. Finally, they pick the location of the closest landmark and assign its location to the target IP.

To evaluate WebCBG, the authors use three ground truth datasets consisting of 88 US academic IPs, 72 IPs from US residential ISPs, and an unknown number of IPs from an online US driving directions website. They obtain remarkable median error results of 0.69, 2.25, and 2.11 kilometers. Unfortunately, this approach has several downsides. First, it does not work for IP addresses where ICMP replies are disabled or filtered out by upstream routers. The residential IPs ground truth set starts out at 241 IPs, but in the end gets dwindled down to only 72 IPs, partially due to this problem. Second, it works poorly in areas of low population density. As population density decreases below 100 people per square mile, the error distance surpasses 10 kilometers and keeps increasing. Third, as all methods that use network delay, this approach fundamentally does not scale to a large number of IP addresses. Wei et al. have subsequently tried to reproduce the results, but on a ground truth of 160 geographically dispersed servers located in the US they could only achieve a mean error of 14.15 kilometers [98]. Liu et al. have also re-implemented this approach and could only obtain a median error of 7.7 kilometers using Chinese IPs [62].

Wei et al. build upon the WebCBG approach by combining it with location hints extracted from reverse DNS hostnames [98] to obtain **RUEL**, which stands for Recursive Undns Evaluating Landmark Algorithm. They use location hints from the routers found earlier in the traceroute path to validate the location of landmarks further down the path. This additional validation steps reduces median error from 14.15 kilometers to 9.78 kilometers using a ground truth of 160 US IPs.

Backstrom et al. [104] propose an approach which we call **SocialGraph**, that relies on a user's social graph to determine their location. This work is not specifically aimed at improving IP geolocation, and in fact in later steps they use a commercial IP geolocation as a secondary source of user location. They derive the location of target users based on the locations of their friends. They find that at medium to long-range distances, the probability of friendship is roughly proportional to the inverse of distance. But this finding does not apply when there is a short distance between friends. Another finding is that people who live in cities tend to have friends that are more geographically dispersed.

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 commercial IP geolocation baseline to 67.5% for the proposed method. Figure

2 in the paper reveals that the median error is about 14 miles, or 22.5 kilometers. 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. While this method scales well, there are two obvious disadvantages. First, it relies on a large proprietary social graph where users self-report their location. Out of 100 million Facebook users in the US, only 3.5 million self-reported a location that could be successfully geocoded. Second, to successfully predict locations of users that have not provided a location, they need to be connected to a relatively large number of friends with known location.

There is a long line of other similar research that aims to determine user location, as opposed to IP geolocation, by mining the contents of their social posts [105–114]. For example, Cheng et al. [105] show that they can geolocate 51% of Twitter users within 161 kilometers of their actual location, using only the textual contents of their posts. However, their ground truth set contains only 5,119 users and their average error is 2,853 kilometers.

In **Checkin-Geo** [62], Liu et al. use checkins logged by a location sharing social network for IP geolocation. Their approach uses data from an unnamed Chinese social network where users can *publicly* check-in to their home or office addresses. This feature is not customary in other social networks with checkins, such as Foursquare. The authors first map users to their home and office locations by clustering all their checkins and looking for patterns such as hours of the day, and counting instances of checkins from the same location. They separately map the same users to their home and office IPs, by mining the server logs of the social network and looking for patterns. They then combine these two mappings to obtain locations for IP addresses. They also propagate the locations from mobile phone IPs to locations of Desktop PCs, through shared user accounts. Finally, they apply IP location interpolation to expand IP coverage.

The best variant of their approach achieve a median error of only 799 meters on 243 IPs from Shanghai. A second larger experiment on 5000 Tencent users in China resulted in a median error of 2.5 kilometers. While the results are impressive, this technique has a significant limitation. It assumes location sharing networks allow public checkins to home addresses, which is typically not the case outside of this very specific case. It also assumes access to a large amount of data in the form of checkins and server logs. Finally, it is unclear why they could not directly correlate the individual checkins with IPs, since they had access to the backend logs.

Early work by Moore et al. on **NetGeo** [115] uses WHOIS databases to obtain IP geolocation information. These databases are maintained by the five Regional Internet Registries, which allocate IP addresses to organizations. The authors highlight the difficulty of parsing different types of WHOIS records and extracting locations. However, they unfortunately provide no evaluation of this method.

Endo and Sadok [116] also extract location information from these databases and develop **WBG**. They query for WHOIS records of target IP addresses and of their corresponding Autonomous System network numbers. As an additional heuristic, they also use traceroute information to find the location of the closest router to a target IP address. WBG achieves an accuracy of ≈52% at the city level on a ground truth dataset of unknown size from a Brazilian ISP.

Li et al. substitute the ping network latency used in GeoPing with HTTP latency obtained from HTTP get requests to obtain a method called **GeoGet** [117]. They direct each target IP to measure latency when performing GET requests to their servers. They then assign to the target IP the location of the server with lowest measured latency. They obtain a median error of 120 kilometers, using a ground truth set of 424 Chinese IPs. Unfortunately, this approach has a couple of disadvantages. First, they assume that the target IPs themselves can be controlled to perform measurements to the servers. Second, their ground truth set is heavily biased since they only retain IP addresses where multiple commercial databases agree on the city. This filtering can cause the ground truth set to be skewed towards IP addresses for which geolocation is easier. Also, previous research has shown that commercial databases are not always accurate [27].

Lee et al. combine self-reported location data from a Korean crowd-sourced broadband speed test with IP location interpolation to assemble a detailed geolocation database we named **Speed** [59]. They use 32 million speed test records from 9 million unique IP addresses, gathered over a period of 7 years. These records were collected through Speed [118], a website operated by the

National Information Society Agency of Korea. Each speed test collected the IP address of the user, along with self-reported location. To increase IP coverage, they perform interpolation using a majority rule vote with a threshold of 80% to assign individual IP address locations to entire IP ranges. Unfortunately, this approach does not scale to other countries since the dataset is specific to Korea. Also, the paper does not evaluate the technique against ground truth data. Instead, they only determine the level of agreement to two commercial databases. They obtain a distance difference smaller than 50 kilometers for 55% of cases when compared to MaxMind, and 53% when compared to IP2Location.

5.5 Limitations of Previous IP Geolocation Approaches

This section presents the limitations and trade-offs of past work in IP geolocation. We also put our research in context and highlight the differences between our approaches and the ones in prior research. To aid in this discussion, please refer to **Table 5.1**, which presents the categories of IP geolocation approaches explored by previous work, and to **Table 5.2**, which summarizes the evaluation results across the papers covered by the previous sections. In these tables, the *traceroute* category is separate from the *ping* category, although both network tools use ICMP packets. We made this distinction because in addition to latency, traceroute approaches also use hop counts, or they use the intermediary nodes on the traceroute paths.

Previous IP geolocation research suffers from a variety of problems that spans multiple types of approaches, which we summarize below:

- Small ground truth set [29, 39, 70–74, 77–79, 85, 86, 91, 92, 94, 95, 98, 103, 116, 117]. The majority of past IP geolocation approaches have only been evaluated against a small number of IPs, on the order of tens of hundreds of addresses. Considering there are 3.7 billion usable IPv4 addresses, and more IPv6 addresses, geolocation approaches should be evaluated against bigger ground truth sets.

- Insufficient geographic diversity [28–30, 59, 62, 70, 71, 74, 79, 85, 86, 91, 93–95, 98, 103, 104, 116, 117]. IP geolocation approaches need to cover the entire world. However, many geolocation techniques we discussed in the previous sections are either designed to work in a specific geographic region, or they are only evaluated in a single country with high Internet penetration.

- Lack of networking environment diversity [28, 29, 39, 72–74, 77–80, 85, 86, 91, 92, 92–95, 98, 100, 117, 119]. Several techniques are designed and evaluated in specific networking environments: they use IP addresses in academic institutions or ISP backbones with very good terrestrial Internet connectivity and low latency, they use IP address datasets filtered to only contain router addresses, or they only use addresses which respond to ICMP packets. Large scale IP geolocation methods should ideally be able to handle more diverse environments, including last-mile scenarios with high latency caused by end-user cable or DSL connections, or cases where ICMP packets are filtered out at network borders.

- Poor city-level accuracy [28, 39, 70–74, 85, 86, 91–95, 117]. Much of the past research in this area features error distance results on the order of 40+ kilometers, or often even hundreds of kilometers. Using these approaches for city-level accuracy, which is the granularity targeted by commercial IP geolocation databases, is not possible. Table 5.2 lists previous evaluation results.

- Non-existent or vague evaluation [87, 96, 102, 115, 120–123]. These approaches either do not have any kind of evaluation on against a ground truth set (for example HawkEyes [120] has no evaluation), or have vague or insufficient evaluation (for example HLOC [102] is compared to DRoP by using only agreement statistics).

- Usage of commercial IP geolocation databases for training or testing [59, 93, 104, 117, 124]. Multiple papers use commercial geolocation databases for either training or testing. Since the methods used by these databases are proprietary, and since their terms of service specifically prohibit publishing comparative evaluations using their real names, it is not appropriate to use them as the main data source for either purpose. For example, GeoGet [117] uses multiple commercial databases for training, and NB-LHP [93] uses the MaxMind commercial database for testing.

In our work we purposely rely less on network delay approaches and more on web mining techniques such as mining query logs and extracting locations from reverse DNS hostnames. We have seen from prior work that these types of approaches typically have higher accuracy and coverage, as evidenced by Table 5.2.

In this dissertation we address several of the limitations in prior IP geolocation work:

- Large ground truth sets. We use ground truth sets of millions of IP addresses, spanning all the countries in the world. In Chapters 3, 4, and 7 we use a ground truth set of 8.4 million IP addresses, while in Chapters 8, 9, 10, 11, 12, 13, and 14 we use a ground truth set of 70 million IP addresses. These two ground truth sets are by far the biggest ones reported in literature. Furthermore, in Chapter 12 we use data from PeeringDB to compile a ground truth spanning 400 IP ranges, which we are making public.

- Diverse Evaluation. We perform our evaluation on IP addresses that span all types of networking environments and all types of end user clients, including backbone routers, fiber connections, cable, DSL, and mobile phones.

- High city-level accuracy. We demonstrate median errors below 10 kilometers for multiple of our approaches (see Chapters 9, 10, 11, 12, and 14). These results have high enough accuracy to be used for most IP geolocation applications.

- Good IP coverage. We achieve coverage on the order of millions of IP addresses across each of our proposed techniques, while previous work typically covers hundreds or thousands of addresses. To further increase IP coverage, we use IP location interpolation in Chapter 11, and we conflate our approaches in Chapter 14.

We now turn our discussion to specific types of geolocation approaches. The *ping* and *traceroute* columns in Table 5.1 show that network delay and network topology approaches are the most prevalent in IP geolocation literature. However, both of these categories have significant limitations. First, all such methods require access to servers 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. Using typical network delay approaches to locate the billions of addresses currently in use is not feasible. Third, not all networks allow ICMP pings or disclose their network topology. Network delay based approaches fail in these environments. Fourth, routes on the Internet do not necessarily map to geographic distances. Mobile phone towers, cable routers, and DSL modems can cause much larger latencies than the typical speed-of-light assumption used in network delay approaches. Fifth, the ground truth data for work in this area is usually limited to a few tens or hundreds of IP addresses, typically located in the United States. 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.

Reproducing results of network delay techniques has proven to be difficult. Multiple papers have tried to reproduce the results for CBG [28, 92, 119], TBG [119], Octant [28], WebCBG [62, 98], with little success. Table 5.2 lists these attempts, which invariably obtain higher error rates than the original implementations.

Recently, Ciavarrini et al. have demonstrated that pure network latency approaches such as CBG have a best-case error of 20 kilometers and that obtaining an error below this threshold requires a number of active measurement servers so large as to be unpractical [80]. This means delay based approaches cannot work well in densely populated locations. All of these reasons make network latency and topology approaches difficult to use for practical applications.

The only network latency related approach we investigate uses a traceroute dataset that CAIDA continuously maintains and makes available to researchers [125]. The reason we use this dataset is that it contains a large amount of traceroutes, with the potential to yield high IP coverage.

Instead of pursuing network latency, we mainly study other approaches, including extracting locations from search engine query logs, parsing location hints from reverse DNS hostnames, propagating user IP locations through clicks, IP location interpolation, extracting data from bulk WHOIS databases, and conflating multiple data sources together to improve accuracy and coverage.

In the case of search engine query logs, we mine location hints from the queries issued by users. This approach bypasses the limitations of delay measurements and network topology methods since it does not require active probes. The technique scales with the amount of input information. To the best of our knowledge, query logs have never previously been studied as a source of IP geolocation information.

For parsing reverse DNS hostnames, we use public information including a freely available geographic database and easily obtainable host names. Since hundreds of millions of hostnames contain location hints, it is easy to see why such a large scale method is attractive, over using network delay approaches. Although multiple papers have used reverse DNS hostnames in the past, most of them have used manually generated rules that are labor intensive, limited to only a few popular ISPs, and prone to becoming stale [29, 31, 70, 71, 86, 87, 96, 98, 100, 102, 115, 124]. We demonstrate these problems in Chapter 8. We cast this task as a machine learning problem, where for the first time we train a classifier that can automatically learn how to parse hostnames across a diverse set of domains. Therefore, it does not require manual input and can be re-trained periodically. It also handles unique situations better, since it considers the terms of each hostname individually, without relying on only domain-specific training. In contrast to HostParser, which is a proprietary reverse DNS hostname parser used in Alidade [31], we completely describe our features and make the entire classifier available as open source.

We use clicks obtained by search engine logs for IP geolocation, which is another technique never before studied in geolocation literature. As with query logs, this approach does not require active probes and can scale indefinitely with the size of the logs. We detail two variations, one where we obtain user locations from GPS and propagate them through clicks to users with unknown location, and another one where we remove the dependency on GPS data and instead mine locations from the body of web documents.

We also extract location information from WHOIS databases. This data covers a large part of the IP space and it can be downloaded in bulk from the

Regional Internet Registries. Some limited work in this areas exists [29, 31, 115, 116, 124], but it does not evaluate the accuracy of WHOIS location data in detail. We expand upon prior work by characterizing WHOIS data in more detail, and by evaluating it across four continents.

Finally, here we introduce the novel concept of geolocation conflation, which allows combining any number of disparate IP geolocation sources with different confidence ranges, in order to increase both accuracy and IP coverage.

Table 5.1: Summary of IP geolocation approaches

Year	System	Ping	Trace-route	Reverse DNS	Inter-polation	WHOIS	BGP AS	Crowd So-cial	Other
2000	**NetGeo** [115]			✓		✓			✓
2001	**GeoTrack** [70, 71]		✓	✓					
2001	**GeoPing** [70, 71]	✓							
2001	**GeoCluster** [70, 71]				✓		✓		
2002	**undns** [87]			✓					
2004	**Ziviani** [72]	✓							
2004	**CBG** [39, 73]	✓							
2004	**PBE** [85]	✓	✓						
2004	**RBE** [85]	✓	✓						
2006	**TBG** [86]		✓	✓					
2006	**GeoBuD** [92]	✓	✓						
2007	**Octant** [29]	✓	✓	✓		✓			
2009	**SG** [74]	✓							
2009	**Structon** [30]		✓		✓		✓		✓
2010	**SocialGraph** [104]							✓	
2010	**WBG** [116]		✓			✓			
2010	**Geo-RhOL** [91]	✓	✓						
2010	**NB-LHP** [93]	✓	✓						✓
2011	**Spotter** [28]	✓							
2011	**WebCBG** [103]	✓	✓						✓
2011	**HawkEyes** [120]	✓	✓						
2012	**SDP** [77]	✓							
2012	**Posit** [94, 95]	✓	✓						
2013	**GeoGet** [117]								✓
2013	**Checkin-Geo** [62]				✓			✓	✓
2013	**PCFL** [98]	✓	✓						✓
2013	**RUEL** [98]	✓	✓	✓					✓
2013	**PathAudit** [96]			✓					
2013	**AdvancedGeo** [124]		✓	✓		✓			✓
2014	**DRoP** [100]		✓	✓					
2015	**Dragoon** [121–123]	✓	✓						
2015	**Alidade** [31]	✓	✓	✓	✓	✓	✓		✓
2016	**GeoSpeed** [59]				✓			✓	
2016	**AIG** [78]	✓							
2016	**Neural-RBF** [79]	✓							
2017	**HLOC** [102]	✓		✓					
2018	**CRLB** [80]	✓							

Table 5.2: Results summary for network delay and network topology techniques

Method	Ground Truth	Results
GeoTrack [70, 71] ↪ Traceroute ↪ Reverse DNS	**265 IPs** - UnivHosts - manually curated dataset of IPs located on campuses of US universities.	**102 km** median error
	2,380 IP addresses - FooTV - users of an online TV program guide looking up zip codes	**590 km** median error
GeoPing [70, 71] ↪ Ping	**265 IPs** - UnivHosts - manually curated dataset of IPs located on campuses of US universities.	**382 km** median error
GeoCluster [70, 71] ↪ IP Interpolation ↪ BGP/ASN	**265 IPs** - UnivHosts - manually curated dataset of IPs located on campuses of US universities.	**28 km** median error
	181,246 IPs - bCentral - User self-reported zip codes on a business web hosting site	**685 km** median error
Ziviani [72] ↪ Ping	**397 IPs** - *LibWeb* university hots, and hosts part of the RIPE *Test Traffic Measurements* project [68].	**314 km** median error
CBG [39, 73] ↪ Ping	**95 IPs** - NLARN AMP [126] - IPs from USA	**95 km** median error and **182 km** mean error
	42 IPs - RIPE [68] - Mostly located in Europe	**22 km** median error and **78 km** mean error
CBG-Gueye [92] ↪ Ping	**87 IPs** - NLARN AMP [126] - USA	**228 km** median error
	57 IPs - RIPE Atlas [68] - Western Europe	**137 km** median error, 178 km mean error
CBG-Hussain [119] ↪ Ping	**182 IPs** - PlanetLab [75], PingER [84], perfSONAR [127] - Europe, North America, East Asia	**135 - 238 km** mean error, depending on region
CBG-Laki [28] ↪ Ping	**≈100 IPs** - PlanetLab [75]	**175 km** median error
	23,000 IPs - Cogent [76] - USA	**100 km** median error

Continued on next page

Table 5.2 – continued from previous page

Method	Ground Truth	Results
CBG-CRLB [80] ↪ Ping	**95 IPs** - NLARN AMP [126] - IPs from USA	**44 km** median error and **51 km** mean error (These results are the only ones in this table that are **purely theoretical**. They represent the best case theoretical result for *CBG*. See discussion on *CRLB*.)
PBE [85] ↪ Ping ↪ Traceroute	**96 IPs** - From PlanetLab [75] and Scriptroute [128] - at universities	**376 km** median error, ≈1000km - ≈2,400 km mean error, depending on traceroute path length (Figure 5)
RBE [85] ↪ Ping ↪ Traceroute	**96 IPs** - From PlanetLab [75] and Scriptroute [128] - at universities	**346 km** median error, ≈150km - ≈2,400 km mean error, depending on traceroute path length (Figure 6)
TBG-Pure [86] ↪ Traceroute	**10 IPs** - PlanetLab [75] - IPs from USA	**209 km** mean error, with a maximum error of 325 km
	22 IPs - Sprint network - USA	**194 km** mean error
	128 IPs - US universities	**225 km** median error, 253 km mean error
TBG-Passive [86] ↪ Traceroute	**128 IPs** - US universities	**176 km** median error, 178 km mean error
TBG-undns [86] ↪ Traceroute ↪ Reverse DNS	**128 IPs** - US universities	**67 km** median error, 178 km mean error
TBG-Hussain [119] ↪ Traceroute	**182 IPs** - PlanetLab [75], PingER [84], perfSONAR [127] - Europe, North America, East Asia	**127 - 167 km** mean error, depending on region
GeoBuD [92] ↪ Ping ↪ Traceroute	**87 IPs** - NLARN AMP [126] - USA	**144 km** median error
	57 IPs - RIPE Atlas [68] - Western Europe	**100 km** median error
Octant [29] ↪ Ping ↪ Traceroute ↪ Reverse DNS ↪ WHOIS	**104 IPs** - PlanetLab [75] and universities - USA	**≈35.4 to ≈40.2 km** (22 to 25 miles) median error, depending on subset

Continued on next page

Table 5.2 – continued from previous page

Method	**Ground Truth**	**Results**
Octant-Laki [28] ↪ Ping ↪ Traceroute ↪ Reverse DNS ↪ WHOIS	≈**100 IPs** - PlanetLab [75]	**125 km** median error
	23,000 IPs - Cogent [76] - USA	**120 km** median error
SG [74] ↪ Ping	**85 IPs** - PlanetLab [75] - USA	**53 km** median error, 92 km mean error
Structon [30] ↪ Web Crawling ↪ Traceroute ↪ IP Interpolation ↪ BGP AP information	**100 million IPs** in China	**87.4%** city-level accuracy
SocialGraph [104] ↪ Social Graph	**2.9 million users** in USA	**22.5 km** median error
WBG [116] ↪ WHOIS ↪ Traceroute	**Unknown number of IPs** from a single local Brazilian ISP	≈ **52% accuracy** at the city level
Geo-R [91] ↪ Ping	**41 IPs** - Ref-1 - GEANT2 university IPs from Europe	**304 km** mean error
	20 IPs - Ref-2 - subset of Ref-1	**305 km** mean error
Geo-Rh [91] ↪ Ping	**41 IPs** - Ref-1 - GEANT2 university IPs from Europe	**246 km** mean error
	20 IPs - Ref-2 - subset of Ref-1	**251 km** mean error
	151 IPs - PlanetLab [75]	**437 km** mean error
Geo-RhL [91] ↪ Ping ↪ Traceroute	**41 IPs** - Ref-1 - GEANT2 university IPs from Europe	**213 km** mean error
	20 IPs - Ref-2 - subset of Ref-1	**281 km** mean error
Geo-RhO [91] ↪ Ping ↪ Traceroute	**41 IPs** - Ref-1 - GEANT2 university IPs from Europe	**177 km** mean error
	20 IPs - Ref-2 - subset of Ref-1	**156 km** mean error
Geo-RhOL [91] ↪ Ping ↪ Traceroute	**41 IPs** - Ref-1 - GEANT2 university IPs from Europe	**169 km** mean error
	20 IPs - Ref-2 - subset of Ref-1	**149 km** mean error

Continued on next page

Table 5.2 – continued from previous page

Method	Ground Truth	Results
NB-L [93] ↪ Ping	**114,815 IPs** of US routers - Locations from MaxMind DB [22]	**449 km** mean error (278.96 miles)
NB-LP [93] ↪ Ping ↪ Population	**114,815 IPs** of US routers - Locations from MaxMind DB [22]	**446 km** (277.29 miles) mean error
NB-LH [93] ↪ Ping ↪ Traceroute ↪ Population	**114,815 IPs** of US routers - Locations from MaxMind DB [22]	**421 km** (261.89 miles) mean error
NB-LHP [93] ↪ Ping ↪ Traceroute	**114,815 IPs** of US routers - Locations from MaxMind DB [22]	**408 km** (253.34 miles) mean error
Spotter [28] ↪ Ping	≈**100 IPs** - PlanetLab [75]	**75 km** median error
	23,000 IPs - Cogent [76] - USA	**30 km** median error
WebCBG [103] ↪ Ping ↪ Traceroute ↪ Web Crawling	**88 IPs** from PlanetLab [75] in USA	**0.69 km** median error
	72 IPs in US residential networks	**2.25 km** median error
	Unknown IPs from an online US driving directions website	**2.11 km** median error
WebCBG-Wei [98] ↪ Ping ↪ Traceroute ↪ Web Crawling	**160 IPs** in US, 130 from PlanetLab [75] and 30 from Amazon EC2	**14.15 km** median error
WebCBG-Liu [62] ↪ Ping ↪ Traceroute ↪ Web Crawling	**17 IPs** from PlanetLab [75] in China	**7.7 km** median error
SDP [77] ↪ Ping	**81 IPs** PlanetLab [75] IPs in North America	**27 km** median error (17 miles)
	90 IPs PlanetLab [75] IPs in Europe	**42 km** median error (26 miles)
Posit-Hops [94, 95] ↪ Ping hop counts	**283 IPs** in USA - geolocated from DNS LOC records and commercial databases, when they agreed	**314 km** (195.16 miles) median error

Continued on next page

Table 5.2 – continued from previous page

Method	Ground Truth	Results
Posit-RTT [94, 95] ↪ Ping latency	**283 IPs** in USA - geolocated from DNS LOC records and commercial databases, when they agreed	**42 km** (26.15 miles) median error
GeoGet [117] ↪ HTTP latency	**424 IPs** Chinese IPs - geolocated from commercial databases when full agreement	**120 km** median error and 200 km mean error
Checkin-MU [62] ↪ Social checkins ↪ User propagation ↪ IP Interpolation	**243 IPs** from Shanghai, China	**2,829 meters** median error
Checkin-MC [62] ↪ Social checkins ↪ User propagation ↪ IP Interpolation	**243 IPs** from Shanghai, China	**799 meters** median error
	5000 IPs Tencent user IPs from China	**2.5 km** median error
PCFL [98] ↪ Ping ↪ Traceroute ↪ Web Crawling	**160 IPs** in US, 130 from PlanetLab [75] and 30 from Amazon EC2	**10.27 km** median error
RUEL [98] ↪ Ping ↪ Traceroute ↪ Web Crawling ↪ Reverse DNS	**160 IPs** in US, 130 from PlanetLab [75] and 30 from Amazon EC2	**9.78 km** median error
DRoP [100] ↪ Reverse DNS ↪ Traceroute	**16,270 IPs** from routers across 6 ISPs, mostly in USA	**99% true positive rate** for error smaller than 10 kilometers
Alidade [31] ↪ Ping ↪ Traceroute ↪ Reverse DNS ↪ WHOIS ↪ ASN ↪ TCP Latency	**100,000 IPs** from a single European ISP, across 73 cities	≈ **10 km** median error
	882 IPs from Measurement-Lab [90]	≈ **16 km** median error
	66 IPs from CAIDA's Archipelago Measurement Infrastructure (Ark) [67]	≈ **14 km** median error

Continued on next page

Table 5.2 – continued from previous page

Method	Ground Truth	Results
GeoSpeed [59] ↪ Crowd Sourcing ↪ IP Interpolation	**9 million IPs** from Korea	**Unknown** error, 55% agreement with MaxMind and 53% with IP2Location at 50 km distance
AIG [78] ↪ Ping	**89 IPs** in North America - 6 at PingER [84], 52 at PlanetLab [75], 31 at PerfSONAR [127]	≈ **22 km** median error (Figure 6)
	30 IPs in Pakistan - PingER [84]	≈ **7 km** median error (Figure 7)
Neural-MLP [79] ↪ Ping	**1,547 IPs** in US from RIPE Atlas [68], university websites, and local government websites	**5.1 km** median error
Neural-RBF [79] ↪ Ping	**1,547 IPs** in US from RIPE Atlas [68], university websites, and local government websites	**4.1 km** median error

Chapter 6

Privacy and Security

Online privacy is becoming increasingly important. Pew Research has found in 2016 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 [129]. 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 of other people or companies being able to access that information [130]. 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 [131, 132]. They reported that at least 75 companies such as Reveal Mobile [133], SafeGraph [134], Kiip [135], and Fysical [136] 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 [137].

IP geolocation databases provide only coarse city-level or region level location information. Therefore they are more privacy-conscious than the widespread industry practice of requesting exact GPS coordinates through mobile apps or the HTML5 Geolocation API. In the rest of this chapter we will cover the steps we have personally taken to preserve user privacy, and we will summarize research in evading or obfuscating IP geolocation.

6.1 Protecting User Privacy in Our Work

Maintaining user privacy is extremely important. We have designed our geolocation approaches and our evaluation experiments with this sensitive subject in mind.

We have taken steps to maintain user privacy when compiling our ground truth data sets. In Chapters 3, 4, and 7 we use a ground truth set of 8.4 million IP addresses, which is based on GPS location data collected from users that opted in to provide this information. This dataset did not contain any user account information. Location information was only collected when users queried for information in a search engine; we did not continuously track the movements of users. To preserve user privacy, we aggregated and randomized user locations. We first filtered the IP addresses as described in Section 3.1 to discard those which appeared in multiple cities. This ensured that we collected IP addresses that were mostly of fixed broadband connections, or of mobile phones where the users remained within the same city. We then aggregated all locations reported per IP address and we retained only the IPs which appeared with different coordinate readings on at least three separate days in the logs. We implemented this second filter to remove IP addresses where due to operating system bugs the exact same coordinates were reported across several days. In contrast, legitimate location readings always varied slightly, even if by a few centimeters or meters. We then computed the centroid of these locations. Finally, we adjusted the locations in a random direction by 200 meters. As a result, we could not track the movements of users, as we did not distinguish between the users behind the same address and we stored a single final randomized location for each IP address. At no point in this process did we manually access any of the raw location data, and we only used the ground truth set data in aggregate.

In Chapters 11, 8 and 12 we used a second type of ground truth set, covering ≈8.9 million addresses, also obtained from devices with GPS hardware. In Chapters 9, 10, 13, and 14 we again used this second type of ground truth set, but at that later time it contained ≈70 million IP addresses. This second type of dataset was anonymized by an automated pipeline by aggregating all locations reported for an IP address, then adjusting the centroid of each IP address by 584 meters in a random direction[1]. IP addresses with a large variance in reported locations were removed as outliers. We never had access to the raw data. These anonymized coordinates cannot be used to pinpoint individual addresses, but they can be used to locate an IP at a neighborhood level.

We have also taken steps to anonymize our training data. In Chapters 7 and 10 we mine the query logs of the Bing search engine. First, the log is disassociated from specific user accounts. We do not have any access to demographic information about the people that issued the queries. Second, we extract locations from queries using an automated process, and we aggregate all queries by IP range. Therefore, it is impossible to pinpoint details about any particular individual. In Chapter 8, we extract location hints from reverse DNS hostnames. These hostnames are created and made public by Internet Service Providers, and they pose a low risk to privacy. We also aggregate these hostnames by IP range. In Chapter 9, we again mine search engine logs, but we focus on user clicks. Similar to the chapters on query logs, we do not have access to demographic information, and we aggregate locations by IP range. In Chapter 12, we use the *IPv4 Routed /24 Topology Dataset* [125] made available to researchers by the Center for Applied Internet Data Analysis (CAIDA). Like reverse DNS hostnames, traceroute information is generally considered public. Finally, in Chapter 13 we access WHOIS databases from Regional Internet Registries. This information is again made freely available to researchers and it only contains information that was volunteered by the organizations that

[1]We used 584 meters on advice of legal counsel, who suggested this number of meters is sufficient to thoroughly randomize locations to a neighborhood level.

receive IP allocations.

6.2 Evading IP Geolocation

A privacy conscious person might feel uncomfortable with revealing their location, even at the city level of granularity. A natural question then arises if it is possible to evade IP geolocation, in order to maintain privacy.

There is some previous work that analyzes IP geolocation from an adversarial perspective. Muir and Oorschot debate the problem of maintaining privacy in their early survey on IP geolocation [138]. They rank different approaches of IP geolocation by falsifiability, and find that many of them can be mislead to some extent into choosing the wrong location. For instance, information stored in WHOIS records for domains, IP addresses, and Autonomous System networks is to a large degree voluntary. It would be possible to then list incorrect location information. Crowdsourced IP location data is even easier to manipulate, by volunteering false information. Muir and Oorschot also theorize that network delay timings could be modified by a target IP. However, they state that there are some techniques that make it difficult to evade IP geolocation. Analyzing intermediary nodes on a traceroute path can provide clues as to the real location of an IP address. Also, an attacker could use IP interpolation to assign a location to a target IP based on the known locations of other IPs in the same network block. The authors suggest that the best way to circumvent IP geolocation is to use proxies to hide the original IP address. However, they caution that some software can still reveal the original IP address, even when using proxies. They give an example of creating a malicious Java applet that gets loaded from a web page visited by the target user, which then makes an outbound connection and reveals the original IP address.

Gill et al. put into practice the suggestions from Muir and Oorschot by performing experiments to determine if IP geolocation techniques based on network delay, network topology, and reverse DNS can actually be circumvented [139]. For circumventing network delay approaches, their hypothesis is that a privacy conscious user could delay ping replies sent back to active probe servers to make it look like the probed device is further away than in reality. The authors assume that this person is aware of the identities and locations of each of the probe servers, which may not be true in reality. They evaluate this hypothesis on *CBG* [39, 73] (See Section 5.1), using a ground truth of 50 North American IP addresses from PlanetLab [75]. The results show that a target IP can be manipulated to look like it is located thousands of kilometers away from its actual location. They next tested network topology approaches, by fabricating traceroute replies once packets reach one of the servers they control. Due to the way traceroute works, once the packets traverse the actual network and reach one of their servers, that server can from then on send back a fake path, for different TTL values. Using an additional 30 PlanetLab IPs in Europe, they demonstrate faking the location of a target IP by more than 1,000 kilometers. Finally, they also introduce a component designed to misled undns [87], which is a reverse DNS hostname parser. By manipulating both traceroute paths and reverse DNS hostnames, they were able to make it appear as the target IP was located within 50 kilometers of the desired forged location.

More recent work by Abdou et al. [140] further investigates manipulating network delay approaches, but does not address network topology or reverse DNS techniques. To extend the research by Gill et al., they propose four strategies for modeling delay. For instance, one of the variations can only increase ping replies, while another can both increase and decrease the latency of replies. They evaluate evading multiple network delay approaches, including GeoPing [70, 71] and CBG [39, 73]. Experiments carried out on 144 PlanetLab IPs show a reduction of 58% to 83% in median error distance when forging locations, when compared to the experiments by Gill et al. However, like in previous research, this work also assumes the user knows the location of all active probing servers, which is not realistic.

Chapter 7

Improving IP Geolocation Using Query Logs

We investigate the feasibility of using location information extracted from search queries to improve IP geolocation databases at city level granularity. We first present the problem statement, then go over challenges, discuss the approach we have taken, and finally evaluate it against a large ground truth dataset. The aim of the preliminary method presented here is not to optimize the model but rather to determine the predictive value of user queries in determining the location of IP addresses. We further refine using query logs for IP geolocation later on in Chapter 10.

7.1 Problem Statement

In Chapter 4 we evaluated three commercial geolocation databases - Vendors A, B, and C - against a ground truth dataset of 8.4 million IP addresses. These databases have higher accuracy, higher coverage, and lower error distance than most previous research. For instance Vendor B, which had the worst results out of all commercial databases, has a median distance of 18 kilometers in the US, as measured by our ground truth set, while past research presented in Chapter 4 often has a median error of 25 kilometers or more. So even the worst performing commercial database we studied has a much better median distance than most published work. Furthermore, both total IP coverage and ground truth IP coverage are very low in previous research with up to thousands of IP addresses. Comparatively, Vendor *B* has a worldwide coverage of 3.48 billion IP addresses, and in this work we use a ground truth set of 8.4 million IP addresses.

Nevertheless, the accuracy of these commercial databases is still lacking for some practical applications and needs improvement. For example, Figure 4.3 showed that the database provided by Vendor *A* geolocates only 60% of the ground truth IPs within 15 kilometers of their actual location in the United States. If we were to use this database in a search engine setting, it might return incorrect location information for roughly 40% of real user IP addresses. In other countries the error distance can be even higher. If a search engine shows businesses, weather, or news for an incorrect location for 40% of the time, users might switch to a competitor.

To augment existing IP geolocation databases we must introduce new sources of information. One possible solution is to directly use our GPS-based ground truth data. While this approach has potential as we later investigate in Chapters 11 and 14, in this chapter we instead focus on explicit location information extracted from user queries. The reason why we use locations from queries instead of directly using the GPS data is the difference in IP coverage. Although our GPS data covers 8.4 million IP addresses, the information extracted from billions of explicit location queries has the potential to cover many more IP addresses.

Given a search engine query log and a commercial IP geolocation database, our task is to improve the accuracy of the geolocation database using the information in the log by correcting some of the locations in the database at the city level. The implication of starting from an existing database is that we are relying on the existing IP ranges already present in the database. The *maximum* IP range size in the three commercial databases we use is 256 addresses per range, which is a reasonable granularity. Furthermore, the coverage of the starting databases is very high, at more than 3.48 billion addresses each.

In this chapter our goal is to improve existing commercial databases using data from query logs, instead of creating a new database

from scratch. The reason is that while the coverage of the locations we extract from user queries is high at 360 million distinct IP addresses, it is still not enough to cover all of the used IP range space. Later in Chapters 10 and 14 we combine query logs with other data sources and evaluate the result directly against commercial geolocation services.

7.2 Technical Challenges

In order to extract locations from queries and combine them with an existing geolocation database, we had to overcome several technical challenges:

1. **We must extract locations from queries**. Given a query log, we must extract the explicit locations mentioned in the queries.

2. **Query logs have a large size and extracting locations from queries is computationally expensive at scale.** Our query logs contain billions of data points, which makes it difficult or impossible to store and process the data on a single machine. A secondary problem is that extracting locations from queries is computationally expensive. Therefore, we must extract queries which are most likely to contain locations.

3. **The locations in the database and the locations extracted from queries are not normalized.** We must normalize locations to bring them into a common space before we can combine and compare them directly. For instance the query "plumbers in nyc" contains the location "nyc", which cannot be matched directly with the location "40.7141667, -74.0063889" stored in the geolocation database.

4. **For a given IP range there can be multiple candidate locations extracted from the query log.** Users within a certain IP range are likely to search for a variety of different locations. We must rank these locations in order to pick the most likely candidate for the IP range.

5. **We need to compensate for the effect that primate cities have on surrounding towns.** Primate cities can skew results due to their overall popularity. Consider the example in Figure 7.1. Here we plot all the locations mentioned in user queries issued from a single IP range, within one

month. In this example the radius of the circles depicts the number of times each location was mentioned by users. We can observe that London and Reading are both popular, but London is mentioned more often. The actual number of mentions is 2,159 for London and 1,254 for Reading. In reality, the correct location of users in this IP range is Reading. Here we observe that primate cities can skew the number of mentions due to their global popularity. Therefore, in our approach we must correct for this effect.

6. **We must combine the locations extracted from query logs with the pre-existing IP geolocation database**. For a given IP range, whenever a location extracted from queries does not match the location in the geolocation database, we must find a scoring system to choose whether to keep the original location or change it to the one extracted from user queries.

7. **The resulting combined database needs to be evaluated on a ground truth dataset and ideally tested on a production application**. Here the challenge is devising the ground truth set and determining which metrics to use. Note we have already covered the creation of the ground truth set in Section 3.1. It is also worthwhile to obtain more proof of the improvements using a real production application. The challenge in this second case is determining objective measures of user satisfaction.

Since we are focused on the feasibility of using query logs to improve geolocation, wherever possible we have tried to reuse existing state of the art technologies, allowing us to focus on the novel contributions instead of re-implementing solutions to already solved problems. More specifically, our contributions are focused on challenges *4*, *5*, *6*, *7*, and the second part of challenge *2*.

7.3 Datasets

Below is a list of the datasets we use in this section:

- **Main query log**: It contains 180 days of Bing query logs, ending on October 9th, 2014. This dataset spans hundreds of billions of queries.

- **Validation query log**: For parameter tuning we use 30 days of Bing query logs collected prior to the main query log. There is no overlap between the

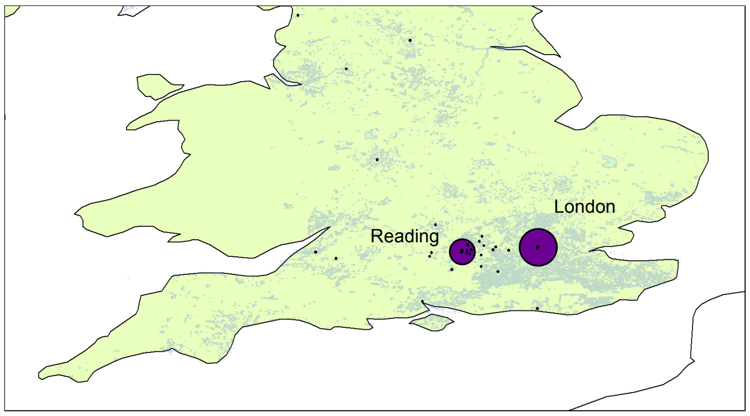


Figure 7.1: An example of the effect of *primate cities*. The map shows locations mentioned in the queries issued from a particular IP range that is located in Reading, UK. The radius of the points shows how often each location is mentioned by the users. If we were to choose the location based solely on number of mentions, we would choose London. In reality, the correct location is Reading.

main query log and the validation log.

- **Baselines**: We use the three IP geolocation databases we previously described in Section 4.2 as baselines. We consider these databases to be the state of the art in the industry.

- **Ground truth**: The ground truth, which we described in Section 3.1, contains 8.4 million IP addresses with known location.

7.4 Approach

We propose improving IP geolocation databases by correcting the location of certain IP ranges using cities extracted from user queries. We begin from the assumption that when search engine users use explicit locations in their queries, in aggregate these queries reveal the users' location. This assumption may not be true at an individual user level. For example, a person might live in New York City, but they might be planning a vacation in Italy. Queries such as "restaurants in venice" might lead to the incorrect conclusion that the user lives in Italy. However, our hypothesis is that if we aggregate queries from users within the same IP range, the locations which are most mentioned will be geographically close to the users in the range. Furthermore, even at the individual level users do not search for the same non-local locations for extended periods of time. In our example, once the user has completed their research on Venice, they might resume searching for locations closer to home.

Users include explicit locations in their queries for several reasons. First, it is possible they do this because of habit or because they do not realize that search engines know their general location. Second, they might have noticed that the search engine returns incorrect location for their searches, and they are correcting it by explicitly specifying the location. For example, if for the query "weather" the answer shows the weather forecast in an incorrect city, the user is likely to click on a search result instead, switch engines, or requery using the explicit location, such as "weather in seattle". Third, they might be searching for information in a location other than their own city. The first two cases yield the true location of the user, while the last case can lead to false positives.

Figure 7.2 illustrates the intuition behind our approach with two examples. In each of the examples we plot the locations mentioned in queries issued from one IP range in a one month interval. The blue dots represent the distinct locations mentioned in the queries, while the shading around them shows how often each location was mentioned. The white dot reveals the real location of users in the IP range, as given by GPS information. The correct cities are Morelia in Mexico, and Florence in Italy, respectively. In both cases the city with most query mentions is also the correct one.

Given search engine query logs and an IP geolocation database, our goal is to improve the accuracy of the database. Below is a summary of the steps we have taken to solve this problem.

1. **Extract queries** and corresponding IP addresses from query logs.

2. **Filter impressions**, keeping the ones likely to contain locations.

3. **Extract locations** from the queries that remain.

4. **Reverse geocode locations** extracted from queries and locations extracted from the target geolocation database.

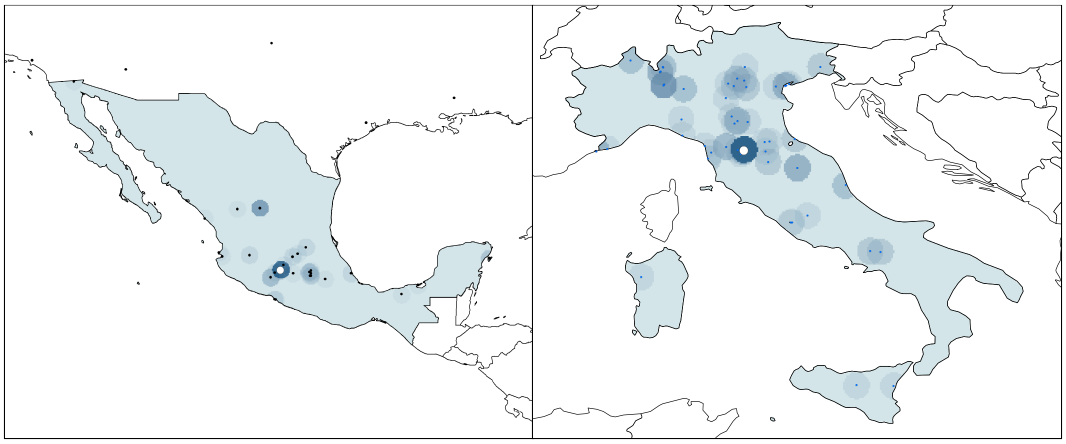


Figure 7.2: Examples of locations extracted from user queries. Each map shows data from a separate IP range. The individual points show the distinct geographical locations mentioned in user queries issued from the IP range. The amount of shading around them shows the frequency at which these locations were mentioned. The larger white dot shows the actual location of users in the IP range as given by realtime GPS data. The maps show that in both cases the location most frequently mentioned in user queries is also the real location of the users. The figure shows Morelia, Mexico on the left, and Florence, Italy on the right. The size of the IP ranges is 60 and 11 IP addresses, respectively.

5. **Aggregate locations** first on IP address, then on the IP ranges in the target IP geolocation database.
6. **Compute the popularity of each distinct location** to be used as a proxy for determining *primate cities*.
7. **Score the location candidates** in each IP range.
8. For each IP range where there are candidates, and where the top query location is different than the original location in the IP range, **decide whether to keep the original location** or modify it based on queries.
9. **Test the modified geolocation database** against the ground truth.

We will now provide details for each of the steps. We first extract the queries and IP addresses from the query log. We have performed this step using an implementation of SCOPE, which is a language for processing massive datasets in parallel across a distributed cluster of machines [141]. Due to the capabilities of SCOPE, this step is trivial to implement. The result is a smaller but still sizable dataset where we have removed unwanted extra metadata.

Second, we filter the impressions in the log obtained in the first step to retain only the queries which are most likely to contain explicit locations. This filtering step is required in order to reduce the number of queries we need to pass to the location extraction phase. We keep the impressions where the query has local intent, such as "plumbers in chicago", using the production Bing query classifier.

Next, we extract explicit locations from queries. To achieve this we use a method similar to the query dominant location extraction algorithm presented in Wang et al. [142]. We use the user query as input. The output contains the detected location, including the country, city, and coordinates. When no location is detected in the query, the output is empty. This results in several billion explicit location queries, issued from 360 million distinct IP addresses. Please refer to Subsection 7.7 for a discussion on reproducing these experiments using publicly available resources.

We then normalize both the locations extracted from user queries and the locations contained in the target IP geolocation database through reverse geocoding using the publicly available Bing reverse geocoding API [69]. Given a latitude and longitude pair as input, the output is a normalized location at the city level which contains the country, state and city. Each city is assigned a unique identifier. Distinct cities with the same name receive different identifiers. Since the geocoding is performed at the city level the public Bing API achieves an excellent reverse geocoding accuracy in all 50 countries we test in this chapter. We perform this step to ensure that the locations extracted from queries and the locations in the target database can be directly compared.

We aggregate the candidate locations per IP address. For each IP address where we have extracted at least one location from user queries, we count the occurrences of each distinct location. We then map these IP addresses on the IP ranges of the target geolocation database, and further aggregate locations per IP range. These two related tasks are performed in a distributed fashion. The result is a list of candidate locations for each IP range, along with their counts.

We then compute the popularity of each distinct city across all IP addresses.

We use these counts as a proxy for determining *primate cities*. We name these counts *GlobalMentions*.

Next, we rank the candidate locations in each IP range to determine the top one for that range. For each location mentioned in an IP range we compute *MentionsNorm*, which stands for *normalized mentions* and is shown in Equation 7.1. For each IP range the candidate cities are ranked in descending order by *MentionsNorm* and the top location is retained. In the equation, *LocalMentions* is given by the number of times the current location was mentioned by users in the current IP range. *GlobalMentions* is the number of times the current location was mentioned across all IP addresses. *IPInst* counts the number of distinct IP addresses in the current IP range that have mentioned the location. Exponents x and y can increase or decrease the importance of *LocalMentions* over *GlobalMentions*. Using a local parameter search on the separate validation dataset we have set x to be 1.5 and y to be 0.5. The parameter search was performed using a step size of 0.1 and limits of 0.0 to 3.0. The exponents can account for *primate cities* by promoting smaller cities and demoting larges ones. In the previous example from Figure 7.1 London will now have a lower score than Reading, which is the correct choice.

$$MentionsNorm = \frac{LocalMentions^x}{GlobalMentions^y} \cdot IPInst \tag{7.1}$$

In the last step, for each IP range where the city in the database does not match the location extracted from user queries, we have to decide if we have to modify the location in the database. To achieve this we introduce Equation 7.2. *IPInstPercentage* in the equation is the percentage of IP addresses in the IP range which mentioned the current location, where *IPInst* is defined as in the previous equation and *EndIP* and *StartIP* are the last and first IP address in the range, respectively.

$$IPInstPercentage = \frac{IPInst}{EndIP - StartIP + 1} \tag{7.2}$$

To determine if we have to perform the replacement we use both equations 7.1 and 7.2. Using the same validation dataset we determine the cutoff thresholds for each equations. That is, we perform the replacement only if the top location in the IP range has *MentionsNorm* of at least 0.3 and *IPInstPercentage* of at least 5%. We have experimentally determined these thresholds

Table 7.1: Summary of evaluation results across three IP geolocation databases and the top 50 countries by IP density. The results are for city level accuracy.

Vendor:	**A**	**B**	**C**
Countries Accuracy Improved	49	49	44
Countries Accuracy Decreased	1	1	6
Median Accuracy Change	+20.7%	+32.2%	+1%
Mean Accuracy Change	+114.8%	+121.1%	+3.4%
Worst Accuracy Change	-0.2%	-0.2%	-0.4%

using a *local parameter search* (see definition in Section 2) with a step size of 0.05 and 0.5%, respectively, a minimum of 0, and a maximum of 1 and 25%, respectively.

7.5 Ground Truth Evaluation

We compared the accuracy and distance error of the three commercial IP geolocation databases to the equivalent databases modified using our approach based on user queries. Figure 7.3 shows the improvements in exact city match accuracy for five high-traffic countries. We observe the best improvement for Vendor *B*, where for countries such as Germany, Italy, and Spain, the accuracy increases by more than 100%. Table 7.1 shows an evaluation summary for the top 50 countries by IP density. For Vendors *A* and *B*, the accuracy improves in 49 out of 50 countries, while for Vendor *B* the accuracy improves in 44 countries. In the few cases where accuracy decreases, it does so by less than 0.4%: in Israel for the first two vendors, and Colombia for the last one. Median and mean accuracy computed across all 50 countries show significant gains, especially for the first two vendors. For Vendor *C* the improvement is more modest, as this baseline had the highest initial accuracy. Nevertheless, for several countries such as India, Belgium, Netherlands, and Mexico the improvements are higher than 5% for Vendor *C*, with Taiwan seeing the best improvement at +74.2%. The reason for the large improvement for Taiwan is that the original accuracy was low, which allowed query based location information to significantly improve accuracy.

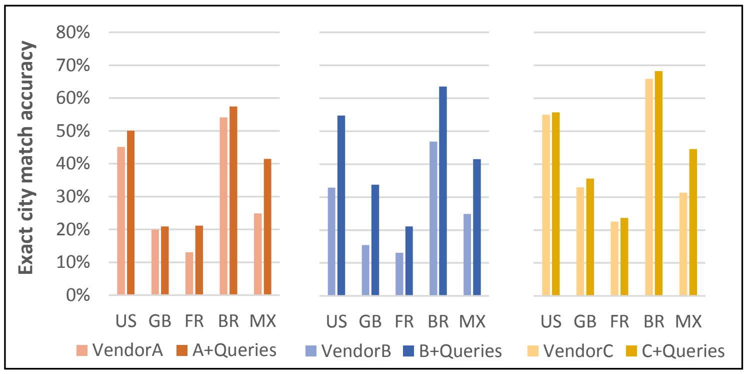


Figure 7.3: Improvement in accuracy for five high-traffic countries between the original IP geolocation databases and the databases modified using locations extracted from user queries.

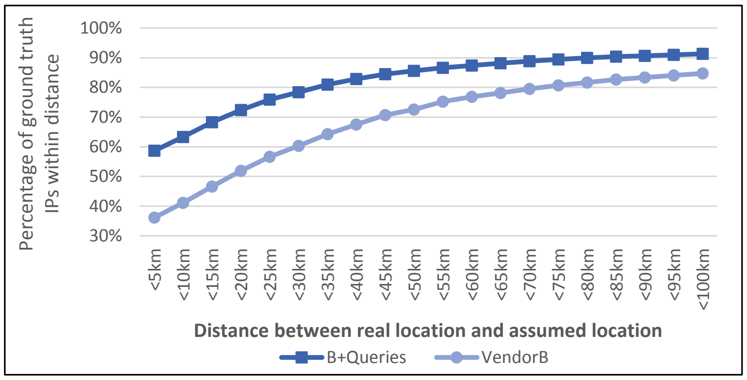


Figure 7.4: Cumulative error distance from ground truth to Vendor B's IP geolocation database and the same database modified using locations extracted from user queries.

The improvements are also apparent when we plot the cumulative error for the distance between correct and assumed locations. In Figure 7.4 we show the cumulative error for Vendor B in the United States. We do not superimpose the results for the other two vendors here to make the figure easy to understand. The shapes of the curves for the other two vendors are similar, but the improvements are less pronounced. The results for Vendor B show a remarkable improvement, as the percentage of ground truth IP addresses where the error is less than 5 kilometers increases from 36.1% to 58.7%.

7.6 Privacy Considerations

Our approach protects the privacy of search engine users, as query logs contain sensitive personal information. First, our method extracts locations from user queries, discarding other words in the queries. Second, all extracted locations are normalized using reverse geocoding. We retain location information only at city level granularity, although some queries initially contain locations which are precise up to the street level. Third, the location information is further aggregated at the IP range level, which combines data from individual users in that IP range into a single set of location counts. Aggregating data across a range of IP addresses makes the data less granular. These steps provide strong privacy safeguards as the output data is coarse and contains no personally identifiable information.

7.7 Reproducing Experiments

Most of the steps in our approach can be reproduced using public techniques and APIs. The IP geolocation databases can be obtained from the companies previously listed in Section 4. Locations can be extracted from queries using publicly available alternatives such as Yahoo! PlaceSpotter [143] and OpenCalais [144]. Another substitute is a Named Entity Recognizer such as Stanford NER [145] or GATE ANNIE [146], combined with a geocoder such as the ones from Bing [44] or Google [147]. The same geocoders also provide functions for reverse geocoding, which would allow normalizing locations. Finally, all distributed aggregation steps can be implemented on an open source implementation of MapReduce, such as Hadoop [148].

7.8 Validating Improvements

In order to validate our IP geolocation improvements, we have carried out an A/B experiment on the Bing search engine during the seven day period ending on November 1st, 2014. The treatment and control variants each spanned approximately 850,000 unique users and 1.6 million queries from the Mexico market, which represents a sizable percentage of that market. For the control we have used a proprietary IP geolocation database obtained by combining the databases from Vendors A, B, and C, and by adding other sources of information. The resulting database has higher accuracy at the city granularity than any of the individual databases from the three vendors. The geolocation database used in the treatment was generated by modifying the location of some of the IP ranges in the proprietary database, using the data and methodology as described in this chapter. Comparing the two databases using the ground truth and methodology explained in Section 4.2 shows that the treatment database has 10% higher accuracy in the Mexico market than the control database. We have purposely chosen to conduct the experiment in a country where our observed improvement is neither the highest nor the lowest.

Table 7.2 contains the statistically significant changes in user metrics computed at the conclusion of the experiment. The first two rows show that both the overall and the answer success clicks have improved, with 0.8% and 1.67%, respectively. By answers we mean the special page blocks, such as restaurant and cinema listings, which are visually different than the algorithmic results. An improvement in the success metrics indicates that users are more likely to be satisfied with the results once they click on a link. The table also shows an improvement in advertising click through rate and click success. These changes can lead to higher advertising revenue. Finally, the metrics show that the click through rate on the Entity Pane has increased by 1.58%. By Entity Pane we mean the right side of the screen, also known as a Knowledge Graph in the context of the Google search engine, which presents rich contextual information such as details about businesses and maps with close-by restaurants. Since a large percentage of such contextual information is based on the location of the user, the increase in engagement could have been caused by displaying more relevant local information.

Finally, we studied the impact of the experiment on the local answer by focusing on the subset of page views from the IP ranges modified by our method. Compared to the control we see that the overall local answer coverage has decreased by 1.65%, but the click through rate on individual items has increased by 75.5%. These results, which are are statistically significant with P-value at 0.04 and 0.03 respectively, show that while the local answer is

Table 7.2: Statistically significant changes in metrics for the A/B experiment carried out on the Mexico market.

Metrics	**Change**	**P-Value**
Overall Click Success	+0.8%	0.03
Answer Click Success	+1.67%	0.04
Ads CTR	+1.57%	0.03
Ads Click Success	+1.57%	0.03
Entity Pane CTR	+1.58%	0.03

shown slightly less often, the engagement with the answer is much higher. The improvement in engagement suggests that the local results might be more accurate in the treatment due to higher geolocation accuracy.

7.9 Discussion and Implications

Our work is the first to propose improving IP geolocation databases using search engine logs. We claim that mining search engine logs is a natural choice for this task, as these logs centralize a great deal of location information from diverse and geographically dispersed users. This approach has several advantages: it does not rely on network delay measurements as previous research, it can scale to cover any country, and it generally leads to accuracy measurements which are higher than previous work. Furthermore, real time GPS location extracted from the logs can be used to generate large scale ground truth data.

Bennett et al. [16] demonstrated that search results can be improved by incorporating location-based features into the ranking function. Here we have shown that incorrect location can impact user experience negatively. Therefore, IP geolocation databases with higher accuracy can result in improvements in location-based personalization. Outside the realm of search engines our work has implications in fields such as credit card fraud protection. We have shown that IP geolocation databases have relatively high accuracy and low error distance in large countries, such as the United States. However, we have also seen that the accuracy is lower in smaller countries or countries with lower Internet penetration. Increasing the accuracy of IP geolocation is crucial to combating credit card fraud in countries such as Ukraine, and Malaysia [3].

Chapter 8

IP Geolocation Through Reverse DNS

8.1 Introduction

This chapter focuses on extracting location information from reverse DNS hostnames assigned to IP addresses. These hostnames can be periodically collected in a short amount of time by performing a reverse DNS lookup for every address in the IP space. For example, the dataset we use in this chapter covers IPv4 addresses and it is refreshed every 7 days. Reverse DNS is the opposite of Forward DNS. Forward DNS starts from a domain or subdomain such as `www.bing.com` and resolves to zero, one, or more IP addresses [149]. Note that multiple subdomains can map to the same IP. Conversely, reverse DNS lookups start from an IP address and typically return zero or one hostnames [150]. Figure 8.2 contains examples of both forward and reverse DNS resolution. 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. In Section 8.3 we discuss reverse DNS hostnames in more detail.

Figure 8.1 exemplifies the information that can be parsed from reverse DNS hostnames. Here we can derive both the location and connection characteristics for the hostname of an IP address. A person reading the name of the hostname can reasonably determine that it references *Wallingford*, a town in Connecticut, USA.

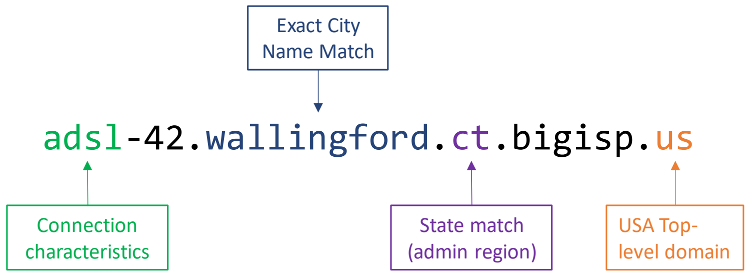
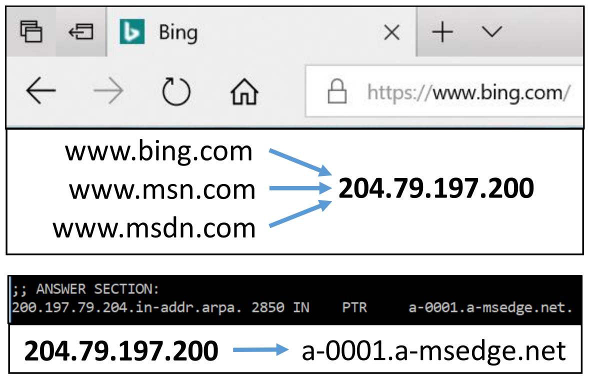

Exact City Name Match
adsl-42.wallingford.ct.bigisp.us
Connection characteristics
State match (admin region)
USA Top-level domain

Figure 8.1: Example of information that can be extracted from reverse DNS hostnames, including location information such as city name, state, country, as well as physical connection characteristics.



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

Figure 8.2: Example of the difference between Forward DNS (top) and Reverse DNS (bottom) resolving. The former starts from a domain and maps to one or more IPs. The latter starts from an IP and maps to zero or one reverse DNS hostnames. Performing a DNS lookup on the `a-0001.a-msedge.net` hostname would most likely, but not necessarily, result in the same IP address. There is no requirement for the reverse DNS hostname to map back to the same IP address.

Given a reverse DNS hostname, our task is to determine its location at the city level. 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 ***New*** *Rich*_**mond**_*, WI* or to ***New*** ***M***_aryla_**_n_**_**d**_*, NB*, or to neither of them? Third, sometimes hostnames contain multiple or 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_.

We propose a systematic approach for using reverse DNS hostnames to geolocate IP addresses. Our contributions are:

1. In our preliminary investigation we determine reverse DNS coverage in the entire IPv4 address space. We also find an upper bound of exact city and airport code matches.

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 potential location candidates, and 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 location popularity.

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 approach as open source. To help the academic community reproduce our results, we release our reverse DNS geolocation software as open source.

8.2 Datasets

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

Our ground truth set contains 67 million IP addresses with known geographic location. To the best of our knowledge, it is the largest and most diverse set used in geolocation literature. We compiled the ground truth set in March 2018 by randomly sampling the query logs of a large-scale commercial search engine. We describe the characteristics of this dataset in more detail in Section 8.5.1.

GeoNames is a free database with geographical information [151]. The March 2018 snapshot we used contains information on 11.5 million geographic features from all countries in the world. From Geonames we used multiple 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 coordinates, original names, ASCII names, alternate names, and codes of administrative divisions. *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. *Admin 1 Codes* is comprised of the codes and names of first-level administrative regions. *Country Info* contains general information about countries, including their Internet top-level domain (TLD).

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 [152] for a token amount[1]. The snapshot contains 24,782 CLLI codes.

[1]As of April 2019, the cost is \$15/month.

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`. This dataset is updated twice a year and it is available for free on the United Nations Economic Commission for Europe website [153]. We used the December 2017 release, which was the latest available version. The dataset contains 108,984 entries.

Public Suffix List, maintained by the Mozilla Foundation, is a free list of domain suffixes under which Internet users can directly register names [154]. Examples include `cloudapp.net` and `gov.uk`. We used a snapshot from February 2018, with 8,066 entries.

Rapid7 Reverse DNS consists of reverse DNS hostnames of the entire IPv4 address space. The dataset is available for free and it is updated weekly. The archive contains snapshots going back to 2013 [155]. We discuss this dataset in detail in the next section.

8.3 Reverse DNS

Forward DNS lookups convert hostnames such as `www.bing.com` into IP addresses, while reverse DNS lookups work in the other direction; they start from an IP address and find a hostname. Since the forward and reverse DNS lookups are set in different DNS records, they do not need to have the same hostname. Reverse hostnames are more likely to be used to name the underlying networking infrastructure, while forward hostnames are used to name websites or other online services [149].

Reverse DNS lookups are achieved by querying DNS records for 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 [150] 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 chapter can be equally applied to IPv6 addresses as well.**

To determine the viability of using reverse DNS hostnames for geolocation, we studied the *Rapid7 reverse DNS dataset* [155], which covers the entire IPv4 address space. Rapid7 compiles it 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 [156]. The preliminary investigation in this section is based on a snapshot taken in January 2017, while starting from Section 8.4 onward we use a more recent dataset from March 2018.

The IPv4 address space consists of all 32-bit numbers. This limits the possible address space to 2^{32} (4.3 billion) addresses. The number of usable IPs is actually only 3.7 billion, since some IP ranges are designated as special-use or private [157]. Not all of these theoretically usable IPs have been allocated yet to organizations. Since not all IP addresses have a reverse DNS hostname, we parsed the Rapid7 dataset to find the actual coverage. We found that 1.25 billion addresses have a reverse DNS hostname. This finding shows that although they have significant coverage, these hostnames need to be augmented with other data to obtain a complete database.

We then quantified how many of the hostnames are valid, since the DNS records are unrestricted strings. We parsed each hostname and rejected the ones that did not respect Internet host naming rules [158]. 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 from the Mozilla Foundation previously described in Section 8.2. This left us with 1.24 billion hostnames, of which 1.15 billion were distinct. Our findings are summarized in Table 8.1, 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.

Table 8.1: Statistics on the usage of reverse DNS hostnames across the entire IPv4 space. More than 1.24 billion IPv4 addresses resolve to valid 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 (naive)	0.16 B	4.4%	14.1%
↪ Airport code match (naive)	0.27 B	7.4%	23.5%

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 positives. One example from the `totbb.net` domain is `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 8.1 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 hostname components of the largest 10 domains in the Rapid7 dataset. We divided each subdomain on the dotted terms, and then we further split the components on dashes and on the transitions between numbers and letters. For example, we split `soc-1.wht2.ocn.ne.jp` into `soc`, `1`, `wht`. We then manually labeled the components that we found to reasonably correspond to geographic locations. We also cross-checked our findings with commercial geolocation databases. Table 8.2 shows a sumary of 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 found 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.*

Table 8.2: 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 based on rows 3 and 5 of Table 8.1.

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**, **mn**
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

Our findings are in line with previous work by Chabarek and Barford [96]. They found that all 8 of the providers they studied used multiple naming schemes. They also found that 20 out of 22 North American providers they surveyed use geographic encodings in their hostnames.

We also studied the distribution of top-level domains (TLDs) 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` 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 potentially be a useful hint in geolocation.

Finally, we compared snapshots of the dataset, each collected in the month of January of years 2014 to 2017, 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 8.3 shows a summary of the results. We found that a maximum of 14.7% of hostnames changed year over year and 63.7% of them remain the same across all four years. These numbers include the cases where one side of the comparison had a hostname but the other side was empty due to the DNS query returning an empty hostname, or 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 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

Table 8.3: Changes in reverse DNS hostnames across 4 years of dataset snapshots

Change / Year Pair	**2014 →2015**	**2015 →2016**	**2016 →2017**
Hostnames changed (incl. empty)	**179M** (14.7%)	**159.3M** (12.6%)	**164.8M** (12.8%)
Hostnames changed (non-empty)	**26.4M** (2.2%)	**20.7M** (1.6%)	**14.2M** (1.1%)
Hostnames gained	**108.5M** (8.9%)	**89.3M** (7.1%)	**81M** (6.3%)
Hostnames lost	**44M** (3.6%)	**49.4M** (3.9%)	**70M** (5.4%)

of hostnames gained every year has steadily declined from 108.5 million in the first pair, to 81 million in the last pair. Conversely, the number of hostnames lost has increased from 44 million to 70 million, respectively. Although there are still more hostnames gained than lost yearly, this gap is narrowing.

In summary, we determined that 1.24 billion IP addresses have valid reverse DNS hostnames with 1.15 billion distinct values, many of which contain exact city or airport code matches.

8.4 Approach

We cast the problem of extracting locations from reverse DNS hostnames as a machine learning problem. We train a binary classifier on a dataset where each training sample is a hostname and location candidate pair, along with a binary label which signifies if the hostname is *likely* or *unlikely* to be in the candidate location. Given a new 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 the classifier, also assigning each candidate a confidence score. For instance, for the hostname `ce-salmor0w03w.cpe.or.portland.bigisp-.net` our approach considers tens of potential location candidates, including *Portland, UK* and *Salmoral, Spain.* In the end however, it ranks *Salem, Oregon* and *Portland, Oregon* as the most likely candidates.

8.4.1 Splitting Hostnames

Drawing from our preliminary analysis in Section 8.3, as well as further manual analysis, we implemented multiple heuristics for splitting hostnames into their constituent components.

First, we apply the *ToUnicode* algorithm described in RFC3490 [159] to convert International Domain Names (IDN) to Unicode. The reason we perform this translation is that international hostnames are stored as ASCII strings using Punycode transcription. For example, the international hostname `xn-0rsod70av79j.xn-j6w193g` gets converted to 夏威夷舞.香港. This allows us to perform location lookups using the original language of the hostname. Second, we separate the subdomain from the domain and the public suffix, using the list provided by the Mozilla Foundation previously described in Section 8.2. These 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. At this point we also extract the native TLD. For instance, for `dps8099.denver.k12.co.us` we extract `denver.k12.co.us` as the domain because `k12.co.us` is a public suffix, we extract `dps8099` as the subdomain, and finally we extract `.us` as the TLD. Third, we split the extracted subdomain 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 8.3 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 terms related to connection characteristics, such as `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 dissertation.

8.4.2 Features

Starting from the results of the hostname splitter, we find the location candidates along with their *primary* and *secondary* features, as defined below. **The**

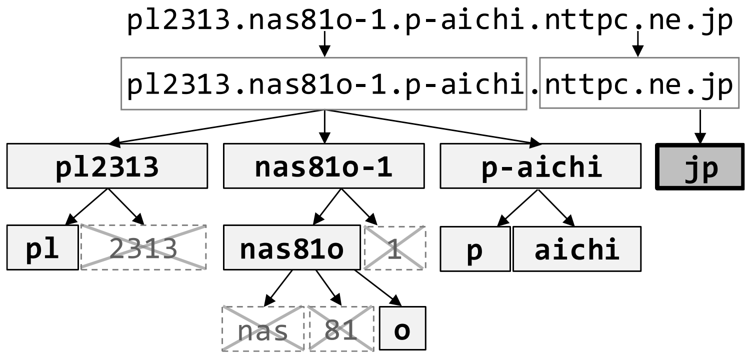


Figure 8.3: Hostname Splitter example with pruning. The figure shows how we progressively split the hostname, first on the domain and subdomain, then on the dotted components of the subdomain, and finally on the individual components whenever there is a transition from letters to numbers or vice versa. The figure also shows how we prune strings which are made up entirely of numbers, or irrelevant networking related words. We separately also extract the top level-domain.

list of preliminary location candidates is defined by the union of locations which match any of the primary features of the hostname. Figure 8.4 shows a concrete example. Primary features can be derived directly from a hostname. These features are matched using a single contiguous string which indicate 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 hostname and location candidate pair. 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 the administrative region for this candidate.

Primary features are based on the *GeoNames*, *UN/LOCODE*, and *CLLI* datasets described in Section 8.2. From *GeoNames* we use the *Cities 1000*, *Alternate Names*, and *Admin 1 Codes* subsets. The primary feature categories are listed in Table 8.4. Each of these categories is represented by three specific features: *IsMatch*, *Population*, and *MatchedLettersCount*. The *IsMatch*

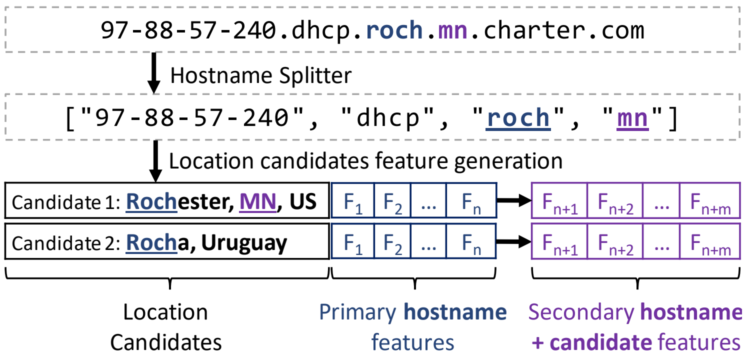

97-88-57-240.dhcp.roch.mn.charter.com
Hostname Splitter
["97-88-57-240", "dhcp", "roch", "mn"]
Location candidates feature generation
Candidate 1: Rochester, MN, US
Candidate 2: Rocha, Uruguay
F1 F2 … Fn
Fn+1 Fn+2 … Fn+m
Location Candidates
Primary hostname features
Secondary hostname + candidate features

Figure 8.4: Feature Matching and Generation. We first split the hostname into its constituent parts. Then, we evaluate each part against a pre-computed a mapping from terms such as `roch` to a list of location candidates and their primary features. In the context of the hostname and each primary feature, we also compute supporting secondary features.

Table 8.4: Examples of primary feature categories. These primary features are used to determine the initial location candidates for every hostname.

Category	**Example**	**Location**
City Name	p907072-li-mobac01.osaka.ocn.ne.jp	Osaka, JP
Alternate names	178235248188.warszawa.vectranet.pl	Warsaw, PL
Abbreviations	cpe-68-173-83-248.nyc.res.rr.com	New York City
City + Admin1	torontoon-rta-1.inhouse.compuserve.com	Toronto, ON
City + Country	er1-ge-7-1.londonuk5.savvis.net	London, UK
No Vowels Name	static-50-47-60-130.sttl.wa.frontiernet.net	Seattle, WA
First Letters	97-90-205-107.dhcp.losa.ca.charter.com	Los Angeles
Airport Code	62.80.122.50.fra.de.eunx.net	Frankfurt, DE
CLLI Code	99-166-111-251.tukrga.sbcglobal.net	Tucker, GA
UN/LOCODE	16.151.88.129,krsel19d.kor.hp.com	Korea, Seoul
Host Patterns	atoulon-651-1-29-109.abo.wanadoo.fr	Toulon, FR

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.

Table 8.5: Secondary feature categories are computed in the context of a hostname and the primary features of a location candidates. Secondary features are used to buttress the location candidates found through the primary features.

Category	**Candidate**	**Match Example**
Admin1	Johnstown, PA	138-207-246-119.jst.pa.atlanticbb.net
First Letters Admin 1	Ft. Huachuca, AZ	frth-bw-noc.ariz.aisco.ngb.army.mil
Country	Paris, FR	ci77.paris12eme.fr.psi.net
Country TLD	Barcelona, ES	barcelona.fib.upc.es

While most feature categories in Table 8.4 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, 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 `rvrs` to *Riverside, CA*. Furthermore, we 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, respectively.

Finally, *Host Patterns* attempts to capture rules not encompassed by the other features. For example, `wanadoo.fr` often prepends the letter *a* to location names, as in `aputeaux` instead of `puteaxu` 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 directly match any location. Using training data we extract frequently co-occurring hostname term permutations of one or two terms. We then aggregate the training data per domain and within a domain on the term permutations. If at least 40% of the training locations for a permutation are located within a 20 kilometer radius, we convert the term permutation into a rule. We determined the support ratio and the distance radius using a small validation set. For example, this feature determines that whenever a hostname in the `frontiernet.net` domain contains the term `or` in the rightmost position and the term `mmvl` in the second rightmost position, then the hostname is most likely located in `McMinnville, Oregon`. This feature would then match for the hostname `static-50-126-80-6.mmvl.or.frontiernet.net`.

To optimize run-time complexity we propose pre-computing 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 pre-compute 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.

We present a possible implementation in Algorithm 1. 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 the 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 pre-computed features for this particular substring and location candidate combination. This implementation allows for fast $O(1)$ lookups at run-time, at the expense of memory usage.

Secondary features are determined in the context of a hostname and

Algorithm 1 Candidates and Primary Features Pre-Computation

```
1: for $c \in PrimaryFeatureCategories$ do
2:     $d \leftarrow Datasets(c)$
3:     for $l \in IterateLocations(d)$ do
4:         $substrings \leftarrow FindSubstrings_c(d, l)$
5:         for $s \in substrings$ do
6:             $f \leftarrow PrecomputeFeatures_c(d, l, s)$
7:             $candidate \leftarrow Candidate(l, f)$
8:             $Features[s] \leftarrow Merge(Features[s], candidate)$
```

location candidate pair. As shown in Figure 8.4, 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 domain *TLD*, respectively.

8.4.3 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 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 8.5 using this classifier.

8.4.4 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 diversity.

We perform stratified sampling on the domain of the hostname, keeping at most $\mathcal{X}$ samples per 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 three parameters in Section 8.5.2.

8.5 Evaluation

We evaluate our approach against three state of the art academic baselines and two commercial geolocation databases. We show that our method significantly outperforms academic baselines and is complementary and competitive to commercial location services.

8.5.1 Ground Truth

Our ground truth dataset contains 67 million 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 a subset of the query log of a major search engine. Each IP address has a corresponding location obtained from users that opted in to provide their location through devices connected to cellular networks or home Wi-Fi networks. We discarded any IP address that was present in multiple cities over the course of a month. The locations were aggregated at IP and city level by an automated pipeline. We did not have access to the locations of individual users.

8.5.2 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 for popularity. We obtained an overall accuracy of 99%, mostly because the vast majority of results were true negatives. 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 8.4.4. 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 exhaustive search but this did not alter the results significantly. In conclusion, our sampling strategy helps the classifier generalize and it significantly improves results.

8.5.3 Academic Baselines

We next evaluate against three state of the art academic baselines. Like 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 [87]. The *DRoP* baseline from CAIDA at University of California-San Diego relies on automatically generated rules derived from hostname patterns and validated by active measurement data (traceroutes) [100]. Finally, the *DDec* baseline also from CAIDA combines the results from *undns* and *DRoP* [101].

Since all three baselines are accessed from a public web endpoint [101], 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 *airtel-broadband.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 selected a list of eight providers, each of which is 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 8.6 lists the results for each of the 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. The first block of results shows median error distance in kilometers. We observe that **our model significantly outperforms the baselines** and its results are generally more stable across all domains. We also observe 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 8.7 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, Italy, 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 some 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, China.* Lastly, since *DDec* is a combination of *undns* and *DRoP*, it is also affected by incorrect rules.

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 far from the actual location, generating larger 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 squared 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 8.6 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.

Table 8.6: Evaluation against three state of the art academic baselines. *undns* from University of Washington consists of manual rules, *DRoP* from University of California's CAIDA uses automatically generated rules, and *DDec* also from CAIDA uses a combination of the first two. Results for our approach, which is fully automated, are under the *RDNS* heading.

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 8.7: Examples of locations extracted incorrectly by the *DRoP* **baseline**, along with the correct location, and the *DRoP* rule that caused the incorrect extraction

Hostname	**Location extracted incorrectly by *DRoP***	**Correct location**	***DRoP* 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.nas931.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`

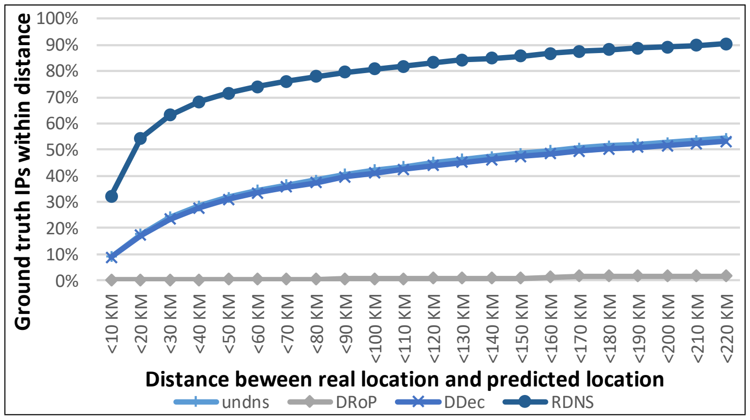


Figure 8.5: Academic Evaluation Error Distance

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 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 8.5 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, significantly underperforming the other methods.

8.5.4 Commercial Baselines

In this work we focus on improving reverse DNS geolocation, which is only one source of geolocation information. Table 8.1 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 completely cover the *IPv4* space.

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

Although reverse DNS geolocation on its own cannot match commercial databases, we evaluate our approach to show that our approach can complement and potentially improve existing databases. We trained our classifier as described in Section 8.5.3. We then obtained two state of the art commercial IP geolocation databases. We tested our approach against the two commercial database providers *A* and *B* using our entire test dataset of 27 million hostnames. The first four graphs in Figure 8.6 show that on certain domains our approach outperforms, and thus can be used to improve, commercial databases. However, as expected, the fifth graph shows that overall the commercial databases still outperform our method. Results show that median error is 43.7, 16.7, 11.1 kilometers, and *RMSE* is 4649, 545.3, 545.9 for *RDNS*, *Provider A*, and *Provider B*, respectively.

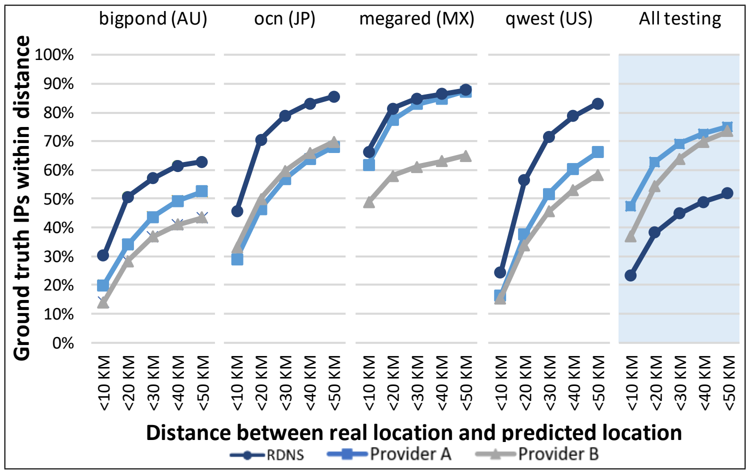


Figure 8.6: Commercial Evaluation Error Distance

8.6 Reproducing Results

To aid in reproducing and extending our results, we are open sourcing all the major components of our approach, including the hostname splitter and the terms blacklist, our sampling strategy, the primary and secondary feature generators, as well as the classifier itself. For feature generation we have purposely used mainly freely available datasets as described in Section 8.2. While we cannot include our ground truth set because it is proprietary, we will make available a binary version of our model. We will also publish instructions on creating a ground truth set using public datasets by using our sampling strategy to minimize any manual labeling.

8.7 Conclusions

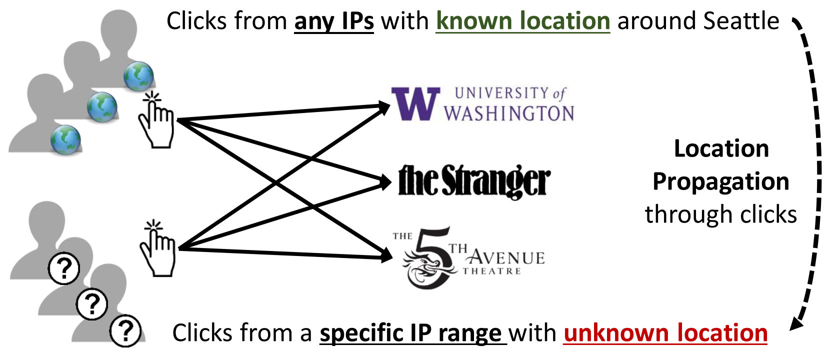
We presented a machine learning approach to geolocating reverse DNS hostnames. Our method significantly outperforms several state of the art academic baselines and it is competitive and complementary with commercial baselines. Our method outputs multiple plausible locations in case of ambiguity. It thus lends itself to being combined with other data sources to form a more complete geolocation database.

Chapter 9

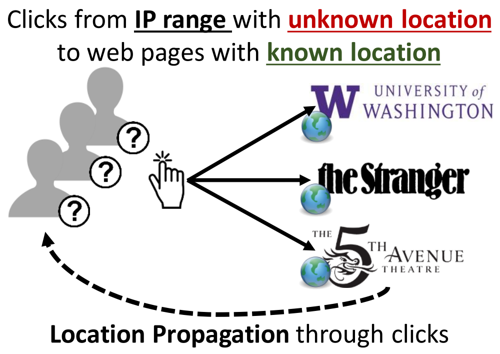
IP Geolocation Through Geographic Clicks

9.1 Introduction

In this chapter we focus on using click logs, in conjunction with a location ground truth set and information mined from web documents, to improve IP geolocation at the city level. We propose first assigning locations to URLs by mining user clicks using two alternative methods. We then propagate these locations to IP ranges with unknown location. Figure 9.1 presents the intuition behind our two proposals. In the first approach, summarized in Figure 9.1(a), we use IPs with known GPS location as the source of geographic information. In the second approach, described in Figure 9.1(b), we mine the web documents themselves and their URLs for location clues. Finally, the second step in both approaches further aggregates locations per IP range, by clustering the coordinates of all clicks from users in each particular IP range. The example in the figures shows that users in a particular IP range often click on URLs that have local affinity to the Seattle area. We posit that the IP range is then also likely to be in the same area.


Clicks from any IPs with known location around Seattle
UNIVERSITY of WASHINGTON
the Stranger
THE 5TH AVENUE THEATRE
Location Propagation through clicks
Clicks from a specific IP range with unknown location

(a) ***GeoClicks-GPS***: 1) Assign locations to URLs by clustering clicks from IPs with known GPS location to each distinct URL. 2) Assign URL locations to IP ranges by clustering all clicks to URLs coming from each distinct IP range. This approach is described in Sections 9.4.1, 9.5.1, and 9.6.1.


Clicks from IP range with unknown location
to web pages with known location
UNIVERSITY of WASHINGTON
the Stranger
THE 5TH AVENUE THEATRE
Location Propagation through clicks

(b) ***GeoClicks-WebIndex*** approach: 1) Assign locations to URLs by mining the body and URL of the clicked documents themselves for location clues. 2) Same step as first method. This approach is described in Sections 9.4.2, 9.5.2, and 9.6.2.

Figure 9.1: Intuitive summary of our two proposed approaches. The difference between the two approaches is that in the first one we derive URL locations from IPs with known GPS coordinates, while in the second we derive URL locations from the body or URL fragments of the clicked web documents themselves. The second step of further aggregating URL locations per IP range is shared by both approaches.

Using click logs to improve geolocation poses several challenges. First, click data is noisy and sometimes contradictory. Users do not always click on URLs related to their immediate vicinity. For example, they may be researching vacation spots, or they may be searching for events in nearby cities. Second, determining the geographical focus of URLs is difficult. Some links can have city-level affinity, while others are more dispersed geographically. Take for instance a regional bank that has branches in three different cities. Furthermore, some websites such as Yahoo Finance have no particular geographical focus, or have only country-level affinity. Our work addresses these challenges by clustering locations at the URL and IP range levels, which reduces noise and outliers. More specifically, our contributions are:

1. In our preliminary investigation we investigate the geographic focus of URLs. We intersect clicks extracted from a search engine query log with a dataset of IPs with known location, to obtain *geographic clicks*. Since the location of these IPs was derived from GPS sensors at the moment of each query, this dataset is not biased by commercial geolocation databases. We then aggregate the clicks by distinct URL and study the characteristics of links with geographic focus, compared to links which are geographically dispersed.

2. We propose a first method to find URLs with local affinity that uses IPs with known GPS location. We use a density-based clustering algorithm to find the location distribution of URLs. We re-rank the clusters for a URL by adjusting their confidence by the prior click density in each area.

3. We also propose an alternate method to find the geographic focus of URLs by extracting location cues from the web documents themselves. While this approach is more noisy and prone to bias, it yields higher IP coverage than the first method.

4. In the second shared step of both approaches we propagate locations from URLs with local affinity to IP ranges with unknown location that have users which also click on such URLs. We intersect a larger click log with the local affinity URLs. We then cluster these URLs per IP range. Finally, for each IP range we weigh the largest centroid by the confidence of the underlying affinity URLs.

5. We evaluate the accuracy of our two approaches against two state of the art commercial geolocation databases. Using a large and diverse ground truth set of 70 million IP addresses with known location, we show that **our approaches significantly outperform two commercial databases on median error, RMSE, and cumulative error distance.** Our method also outperforms prior academic work in both accuracy and scale.

6. Finally, we study the agreement between the two proposed approaches. We first demonstrate that they are complementary and therefore can be used in conjunction. We then show show that there is a high level of agreement between the two methods, and their intersection is highly accurate.

9.2 Datasets

Our ground truth set contains 70 million IP addresses with known location, compiled during the 28-day period ending on October 26th, 2018. To the best of our knowledge, it is the largest and most diverse set used in geolocation literature. It was derived from the query logs of a major commercial search engine from devices with global positioning sensors, where users opted-in to provide location information. The dataset contains both mobile and fixed broadband IP addresses, since users often connect their mobile phones to their home Wi-Fi. It covers the entire world. We never had access to the raw location data. The dataset was anonymized by an automated pipeline by aggregating all locations reported for an IP address, then adjusting the centroid of each IP address by 584 meters in a random direction. IP addresses with a large variance in reported locations were removed as outliers. These anonymized coordinates cannot be used to pinpoint individual addresses, but can locate an IP at a neighborhood level. While throughout this chapter we refer to this location data as derived from *GPS* for succinctness, the dataset actually covers all global positioning systems, including *GPS*, *GLONASS*, *Galileo*, *BeiDou*, etc. [160]. Throughout this chapter we used this ground truth set for both training and testing by performing **ten-fold cross validation**. We randomly split the ground truth set into 10 subsets (folds) of equal size. We then successively trained on nine of the folds, and set aside the last one for testing. We performed this training and testing step 10 times, where each time we used a separate fold for testing. This technique allowed us to test on the entire ground truth set, since we tested each of the ten folds once.

The **GPS clicks dataset** contains 1.1 billion clicks issued from IPs with known location. To obtain it, we first extracted a sample of clicks on any search result page element on the same 28-day period ending on October 26th, 2018. Then, we intersected this data with the ground truth set and only retained the clicks that were issued from IP addresses with known location. The search engine was also aware of the location of users at the time each query was issued initially, therefore this subset of the data is not skewed by IP locations from commercial databases. To also reduce user click frequency bias, we only retained one click per IP per URL in the entire period. For example, the IP of a user clicking on *https://www.miamiherald.com/* thirty times on five different days would only contribute a single click in the dataset. We normalized all URLs by removing the scheme (*http://*, *https://*), the *www.* prefix from hostnames, and the # fragments. For instance, we would normalize the URL *https://www.company.com/About_ Us#Board* to *company.com/About_ Us*. Because this dataset relies on the IPs in the ground truth set, we also segmented this data by the same ten folds.

The **web index locations dataset** contains 4.1 billion distinct web pages with city-level locations extracted from the textual contents of the web pages, or from their URL fragments. We obtained this dataset by randomly sampling from the Bing web index on October 27th, 2018. Each URL in the dataset is mapped to a single primary location. Section 9.4.2 discusses the extraction process in more detail. Locations obtained from the text of web pages pose a low privacy concern since the web pages in the index are public.

The **index location clicks dataset** consists of 2.96 billion clicks issued on URLs with extracted location. To obtain it, we first extracted a sample of search result clicks on the 28-day period ending on October 26th, 2018. Then, we intersected this data with the *web index locations dataset* to retain clicks on URLs with locations extracted from the body of the documents.

The **bulk clicks** dataset contains 14 billion clicks from IPs with unknown location. These clicks were collected from the opt-in logs of a popular browser, and a popular browser add-on, over a three month period ending on October

25th, 2018. To obtain the dataset, we randomly sampled from the impressions which contained an HTTP referer header, which means they were most likely clicks.

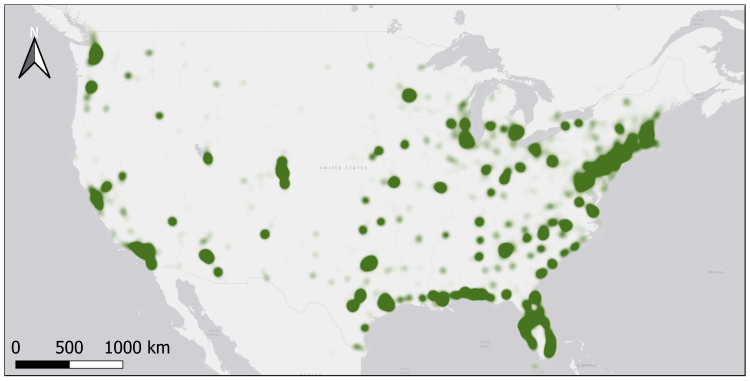
All click logs were anonymized. We did not have access to the identity of users. During our experiments we aggregated clicks at distinct URL level, and then further at IP range level. We never used clicks at an individual user level.

9.3 Geographic Focus

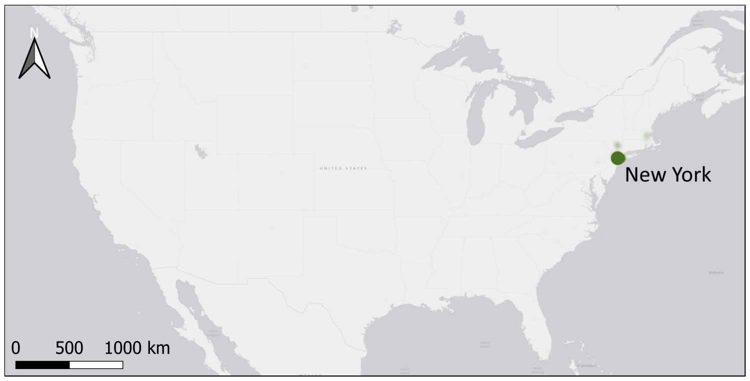
At the onset of our study we set out to determine the viability of assigning locations to URLs using clicks. We also wanted to investigate if it would be enough to assign locations directly to domains, as opposed to individual subpages. As a preliminary analysis we aggregated the 1.1 billion clicks from the *GPS clicks* dataset by distinct URL. We considered the number of click coordinates for each URL as a proxy for their popularity. We then randomly sampled and visualized the coordinates of 100 URLs with varying popularity. Based on our observation, we classify the links into two main categories: URLs which are *geographically dispersed* and URLs which have *local affinity*. We further divide the links with local affinity into *regional*, *local*, and *hyper-local*. The remainder of this section provides details and examples on these types of links.

Figure 9.2 (a) displays a heatmap of the click coordinates on *wunderground.com*, which is a weather forecast website. We consider this URL to be *geographically dispersed*, because its click probability roughly follows the population density of the United States. There is no apparent geographical sensitivity to the coordinates. On the other hand, Figure 9.2 (b) plots a similar coordinates heatmap for *wunderground.com/weather/us/ny/new-york*, which is a specific subpage on the same website. We can immediately recognize that the clicks are concentrated towards the New York City metro area. Based on this example we can draw two conclusions. First, some URLs do indeed show strong local affinity. Second, aggregating clicks only by domain is insufficient. In the case of *wunderground.com*, the domain is *geographically dispersed*, while city-specific subpages exhibit *local affinity*.

To determine which URLs have geographic focus, we first implemented a



(a) Locations of users clicking on *wunderground.com*



(b) Clicks to *wunderground.com/weather/us/ny/new-york*

Figure 9.2: Comparison of a URL which has clicks that are geographically dispersed, with a URL that has clear local affinity.

naïve approach which used a reverse geocoding service to determine the city, state, and country of all coordinates in the sampled URLs. We aggregated the coordinates in each URL by city and sorted by number of occurrences. We then manually visited all the URLs where the top city was present in at least 30% of the clicks. First, we observed that the majority of these links had *local affinity*. Examples include websites for local government and utilities, local businesses such as shopping centers, theaters and concert venues, medical practices, local newspapers and radio, as well as schools and universities. Some of the links are *local* to a city. Figure 9.3 shows that clicks on the website arlington.co.zw were reported within the confines of Harare, the capital of Zimbawe. This website advertises houses for sale in a local gated community. Others are more *regional*. Figure 9.4 displays clicks to bloomsburgfair.com, which is the website of a yearly fair held in Bloomsburg, Pennsylvania. Since the fair draws attention from multiple neighboring counties, it is not possible to assign it a single geographic location. Another similar example of regional focus is bosch.in/careers, which is the careers website for the Bosch company in India. Clicks are concentrated in multiple cities where Bosch has factories or training centers.

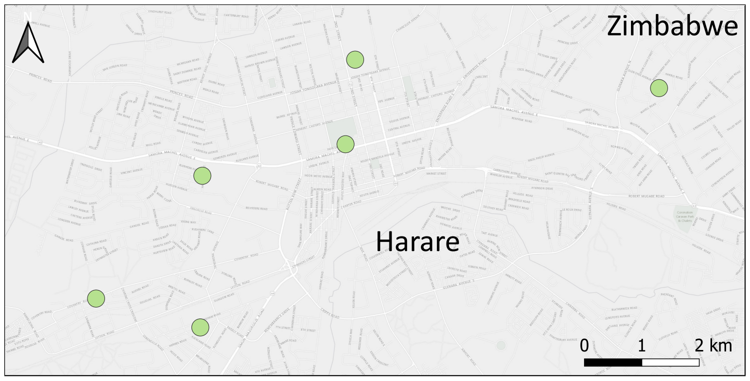


Figure 9.3: Location of clicks to arlington.co.zw, a real estate website selling houses in Harare, Zimbabwe.

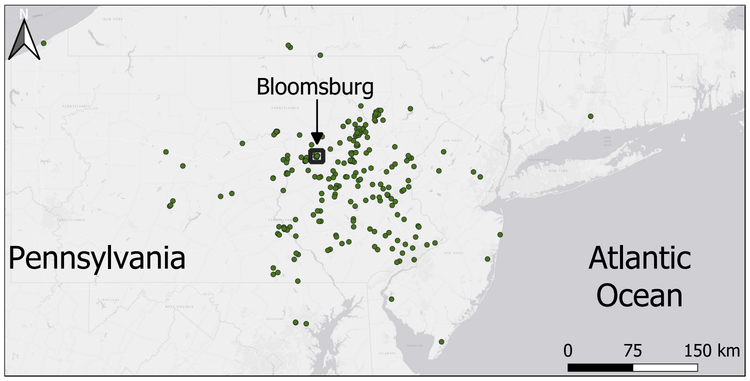


Figure 9.4: Location of clicks to Bloomsburg Fair, a yearly event held in Bloomsburg, Pennsylvania.

URLs can also have *hyper-local* focus. A common example that we observed is student login pages for internal university websites, which are centered on campus locations. But perhaps the most unexpected finding is that there are links that do not have any obvious geographical focus, yet click information shows that they have in fact local affinity. We found more than 1,500 distinct URLs from the *answers.yahoo.com* domain where most of the clicks were

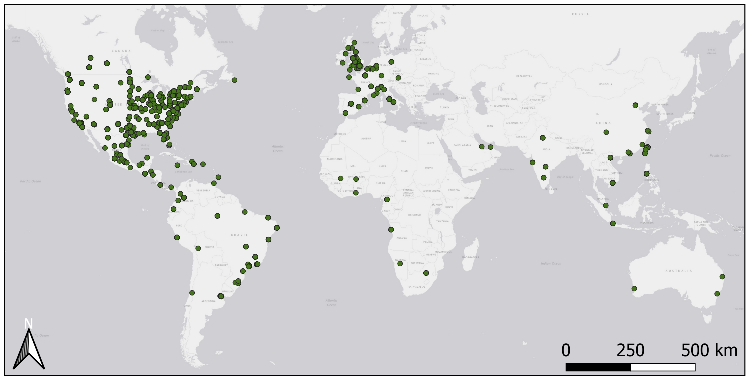


Figure 9.5: Locations of clicks to *Yahoo Answers*. Each point represents the mean location of an individual URL.

within a small radius of a couple of kilometers. Many of these locations were located on campuses of English-speaking universities and schools. Upon studying the content of the pages we determined that the questions on these pages were not related to any particular location but were specific math, physics, and literature homework problems. For instance, one popular question which asks *"Enter the net ionic equation for the reaction of aqueous sodium chloride with aqueous silver nitrate?"* was accessed by 14 different IPs from the campus of a well-known university in Upstate New York. This finding also suggests that some URL locations might also have a temporal aspect. For instance, it is likely the number of these clicks is reduced during the summer holiday. Figure 9.5 shows one mean point for each *Yahoo Answers!* URL that received clicks from at least 5 IP addresses. We note that these types of links are still just a fraction of the 426 million distinct URLs in the *GPS clicks* dataset.

9.4 Assigning Locations to URLs

We propose two methods of assigning geographic focus to URLs. The first method requires access to a seed list of IPs with known GPS location. We aggregate and cluster clicks from these IPs per distinct URL. The advantage of this approach is that, as we will see in Section 9.6, it is very accurate in assigning locations to URLs. The disadvantages are that it requires having access to the coordinates of a subset of IP ranges, and it has low coverage. The second method instead derives locations from the contents of the clicked documents themselves. The advantages of this approach are that it has much higher coverage, and it only requires access to the web index instead of IP location information. However, these advantages are at the expense of lower accuracy. Although both approaches have relatively low median error, in Section 9.6 we will also show that this latter approach has a median error that can be more than double that of the former one.

9.4.1 Locations Extracted from IPs with GPS data

The approach of reverse geocoding coordinates and aggregating by city names in Section 9.3 is not sufficient to find URLs with local affinity. We have found that clicks outside the boundary of a city can also contribute directionally to finding the location of links. For example, a blood bank that serves three neighboring cities arranged in a line receives clicks from all three, but the median location correctly indicates the city in the middle where the organization is located. Furthermore, using reverse geocoding services has its own share of problems and might incorrectly place coordinates in the wrong cities.

Instead of reverse geocoding IP coordinates, we propose using spatial clustering. To better represent geographic focus we use a modified version of DBSCAN [161], which is a density-based clustering algorithm, Intuitively, it groups together coordinates in high-density areas. One feature of this algorithm is that it does not require specifying the number of clusters a priori. Clusters can reach any size as long as they satisfy the density requirements. Another feature is that it can find arbitrarily shaped clusters, which is not possible with other clustering approaches such as the expectation-maximization (EM) algorithm for Gaussian Mixtures. The algorithm has a complexity of $\mathcal{O}(n \log n)$ if the implementation uses an indexing structure for finding neighbors.

DBSCAN requires two parameters, ϵ (epsilon) and *minPoints*. The ϵ parameter represents the radius of the search density range around the current point. If the current point has *minPoints* neighbors which are at most ϵ distance away, then the density bar is met and a cluster is formed. The cluster can grow in any direction and to any size as long as the added points are also in a dense area with at least *minPoints* neighbors. Points in low density areas are considered noise (outliers) and are ignored.

However, using DBSCAN directly yields poor results because of the underlying prior click probability of each geographical area. Cluster sizes are skewed by the presence of *primate cities*. To account for this natural bias, we propose re-ranking clusters. For a URL, given a set of coordinates $G = \{ g_1, g_2, \ldots g_n \}$, DBSCAN partitions G into m clusters, $C = \{ c_1, c_2, \ldots, c_m \}$ clusters, each with one or more points. We define the adjusted confidence of a cluster as its size, divided by the prior probability of clicks on its surface:

$$Confidence(c_i) = \frac{|c_i|}{P(click|Surface(c_i))} \tag{9.1}$$

where we define the surface of a cluster by the polygon which contains all of its points. Note that prior click probability is computed based on the clicks in the entire dataset.

Figure 9.6 shows the click coordinates for the *rosalindfranklin.edu* domain, which is a medical school in North Chicago. DBSCAN extracts two clusters, a larger one with more clicks located in Chicago, and a smaller one with less clicks located in North Chicago, where the school is actually located. Using just the size of each cluster directly would incorrectly lead us to choose the larger cluster. However, re-ranking the clusters by prior probability gives a higher score to the smaller (correct) cluster, since North Chicago has a smaller overall click probability.

After ranking the clusters, we pick the top cluster by score and compute its bounding radius, which is the radius of the circle which encompasses all of its points. We then assign the center (mean) location of the cluster to the URL, if the bounding radius is within a certain threshold as discussed in the Evaluation section. This ensures we retain only URLs which have local affinity.

9.4.2 Locations Extracted from Web Documents

Obtaining a seed list of IPs with known GPS location can be difficult as location from global positioning sensors may only be available to medium and large online services. Since the size of resulting dataset is directly proportional to the size of the seed IP list, a large set of IPs is needed to obtain high URL coverage, which may not always be possible. We describe an alternate

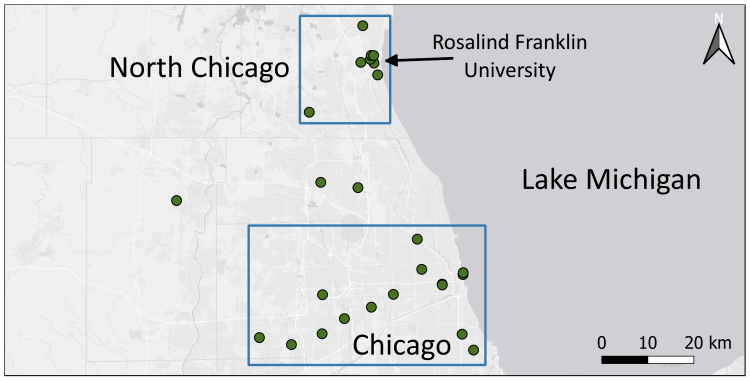


Figure 9.6: Reranking clusters based on region click density. The coordinates shown are clicks on a specific page in the *rosalindfranklin.edu* domain, which is a medical school in North Chicago. Initially, the bottom cluster in Chicago was ranked first. After adjusting the confidence based on prior click density, our approach promoted the cluster in North Chicago to be highest ranked.

approach of assigning geographic focus to URLs that only uses the content of clicked documents themselves, instead of using IPs with GPS location. Our hypothesis is that some web documents contain physical addresses, and these addresses can be later used in aggregate for IP geolocation.

There has been ample prior work on extracting addresses from the body of text documents. Amitay at al. [162] parse web documents to extract a taxonomy of locations using a gazetteer. They report an accuracy of up to 82% on multiple document collections, that together covered 600 pages and 7,000 geotags. Silva et al. similarly use an ontology of geographical concepts to recognize and disambiguate location references, but they also introduce a graph-ranking algorithm similar to PageRank as a second step to further disambiguate locations. They obtain an F-score of up to 0.81 on document collections in 4 languages [163]. Martins et al. take a machine learning approach to this problem by using a Hidden Markov Model learner to find location references, then using an SVM classifier to disambiguate references. They outperforms two state of the art commercial systems [164]. Locations extracted from web documents are typically used to personalize web search [16, 165, 166]. However, these approaches assume user location is already known and correct. The ranking function then finds documents which are close geographically to the user. If user IP geolocation is incorrect, this assumption may lead to irrelevant search results.

To the best of our knowledge, coordinates derived from web documents have never been used to improve IP geolocation. Although the focus of this chapter is not on parsing locations from text but on using them for geolocation, we briefly describe the extraction approach used during web index generation. Locations are found either in the body of the document, or from URL fragments. The parser attempts to locate full postal addresses, zip codes, or mentions of popular cities in the text of documents. For example, the page *nibbanarestaurant.com* is mapped to the coordinates of *Bellevue, WA*, since it is the web page of a local restaurant and the body of the document contains its geographical address. Sometimes URLs also contain location information. For example, weather websites often contain the forecast location in the link. Another example is *redfin.com/zipcode/98033*, a real estate search web page whose URL contains the zip code of *Kirkland, WA*. Each document is mapped to at most a single location. If multiple locations are present in the text, only the first one is used. While this data can be noisy at an individual URL level, we posit that aggregating these locations over millions of clicks can lead to reasonable results.

We first sampled 4.1 billion pages with city-level locations from the index of a large commercial search engine on October 27th, 2018 to obtain the web index locations dataset. We then intersected these pages with a sample of search result clicks on the 28-day period ending on October 26th 2018. We obtained 2.96 billion clicks with locations extracted from web pages.

This alternate approach has a much higher coverage at the URL level of 261 million distinct URLs, as compared to the IP GPS location seed list approach which only yields 3.4 million distinct URLs. However, this second method may introduce higher noise because the locations listed in text documents may not be representative of the locations of users clicking those documents. We further explore this difference in coverage and accuracy in the Evaluation section.

9.5 IP Range Geolocation

In the previous section we presented two approaches to assign locations to distinct URLs. Here we further propagate these locations to IP ranges with unknown location using the separate bulk clicks dataset. **Our goal is to determine a single location per IP range** at the city level, which is the same granularity used by commercial geolocation services. To match the typical layout of these services, we segment the IPv4 space into contiguous ranges of 256 IP addresses (*/24* netmask). For example, the *131.107.174.0/24* range starts with address 131.107.174.0 and ends with address 131.107.174.255. Although we evaluate our approach on IPv4, all methods described in this chapter can be equally applied to IPv6 IPs.

We begin by describing this step using the URL locations data derived from IP GPS data discussed in Section 9.4.1, then later we detail the same step for the alternate approach using the web index discussed in Section 9.4.2.

9.5.1 Using URL Locations from GPS Coordinates

We first intersect the clicks from the bulk clicks dataset with the URLs with assigned location we found in Section Section 9.4.1. The resulting subset contains only clicks to URLs that we previously determined have a certain local affinity. Similar to the previous section, to reduce bias in the data we count clicks from an IP to a URL a single time in the 3 month period. Then, we aggregate the locations of these clicked URLs per IP range. So for each separate range of 256 IP addresses we now have a list of coordinates, where each coordinate is derived from the location of the underlying URLs that have local affinity. Finally, we run DBSCAN on the coordinates in each IP range to determine their predominant locations.

We propose a second method to improve the output from DBSCAN, this time at the IP range level. Given an IP range and its top location cluster, the coordinates that make up the cluster are **each** derived from the location of a single URL. For each of these URLs we have previously computed a confidence score in Section 9.4.1. As the score increases we are more confident that the URL has affinity to that location. Using these scores, we propose adjusting the centroid of the DBSCAN cluster using a weighted average. Since all of the clusters we extract have a small radius of a few kilometers, we can ignore the curvature of the Earth. In the next section, we will demonstrate that this proposal results in a noticeable improvement in distance error.

9.5.2 Using URL Locations from Web Documents

We also perform the same IP range clustering step on locations extracted from the body of web documents. The implementation is very similar to the approach we just took on locations from IPs with GPS coordinates. The main difference is that for the web index data we extracted at most a single location per URL. Therefore, the location for each URL has a confidence of 1. In this case, it is unnecessary to use the DBSCAN weighing scheme and we can directly use the standard DBSCAN output. This alternate method has 13 times higher coverage than the IP GPS method, which leads to more IP range clusters and therefore higher IP coverage.

9.6 Evaluation

To evaluate our two approaches, we first discuss tuning model parameters for each proposal separately. We then compare the proposals against three baselines: two state of the art commercial geolocation databases and a strong academic baseline.

9.6.1 Model Parameters for GPS Locations Approach

We begin by discussing the three parameters we use for the model based on IP GPS locations: ϵ (epsilon), *minPoints*, and the maximum cluster bounding radius. We show that by filtering on the bounding radius of the output clusters in the first step we can obtain a desired balance of accuracy and IP coverage in the second step.

Our geolocation approach consists of two DBSCAN clustering steps. In the first step we cluster locations at the URL level, and in the second step we further cluster URL locations at the IP range level. We run the clustering algorithm separately for each URL and then separately for each IP range.

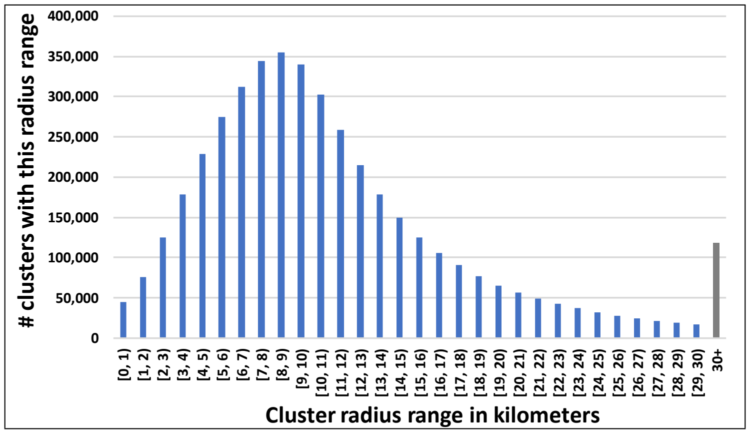


Figure 9.7: Distribution of bounding radius for clusters extracted from URL click locations for $\epsilon = 16$, $minPts = 5\%$. The figure shows a normal distribution centered around the [8, 9) data point, which shows that there were about 355,405 clusters with radius between 8 and 9 kilometers. The long tail demonstrates that DBSCAN can generate clusters of dramatically different sizes, as long as the underlying coordinates abide by the density criteria.

DBSCAN requires two parameters, ϵ and *minPoints*. To find the optimal values for our task, we experimented using a separate validation set of 3 million IP addresses. We set ϵ to 16 kilometers (10 miles) for both clustering steps. This parameter does **not** represent our desired cluster radius, but it represents the neighboring density threshold. DBSCAN can find clusters of any size, as long as the density requirements are met. Figure 9.7 helps demonstrate this property of DBSCAN. The long tail of the figure shows that the output clusters can sometimes cover a large surface, as long as the points are dense. We initially set the second parameter *minPoints* to a fixed size, but we soon discovered that we obtained better results if we assigned it dynamically to be 5% of the number of input points. So, for instance, if a URL contained 100 click coordinates, we set *minPoints* to 5.

To compute the confidence score for each URL (Equation 9.1), we approximate the prior click density in an area by using Geohash [167, 168], which is a well-known geocoding system for latitude and longitude. We aggregate all coordinates in the GPS clicks dataset by Geohash ID. We set the Geohash precision to 5 characters, which divides the entire world in 4.9km by 4.9km tiles. In each tile we count how often we observe location clicks across the entire dataset. This allows us to determine a rough prior click probability for any location in the world, by consulting the density in its equivalent geohash tile. After re-ranking the clusters by confidence, we pick the cluster with the highest score.

In addition to the ϵ and *minPoints* parameters, we also set a maximum bounding radius for the clusters generated in the first step. The bounding radius of the cluster is determined by the circle which encompasses all points in the cluster. **By filtering on the bounding radius at the end of the first step, we can tune the amount of accuracy and IP coverage we eventually achieve in the second step.** Figure 9.8 demonstrates the effect that varying this parameter has on both median distance error and IP coverage. We define distance error as the distance between where a model places the coordinates of an IP, and the actual location of the IP as given by our ground truth. We define IP coverage as the percentage of IPs from the ground truth set for which a model makes a decision.

To further show the effect of tuning parameters for accuracy or coverage, we will evaluate two instances of our model based on IP GPS data, *GeoClicks-GPS-HigherAcc* and *GeoClicks-GPS-HigherCov*. For the former we set the maximum bounding radius to 6, while for the latter we set it to 20. One version is tuned for higher accuracy, while the other one tuned for higher coverage. *HigherAcc* has an IP coverage of 2.24%, while *HigherCov* has a coverage of 52.21%.

9.6.2 Model Parameters for Web Index Locations Approach

The alternate method to assign geographic to focus to URLs makes use of locations extracted from the text of web documents. Since in this approach we do not have to cluster IP coordinates, in the first step we directly assign at most one location to each URL. In the second step we aggregate and cluster URL locations based on clicks issued by users in each IP range. This second step allows us to tune the DBSCAN clustering parameters for higher accuracy or coverage.

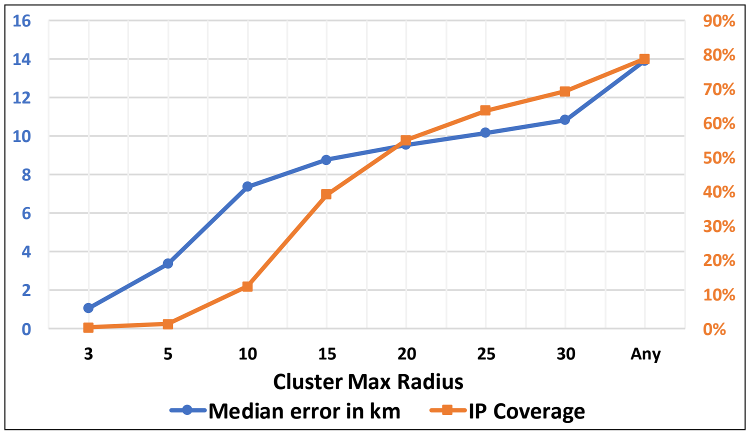


Figure 9.8: Effect of varying cluster max radius on median error and on IP coverage. As we increase the maximum radius parameter, both the median error and the IP coverage increase. This setting allows selecting a balance of accuracy and coverage.

9.6.3 Commercial Baselines

We compare the two instances of our model to two state of the art commercial databases, one labeled *ProviderA* and the other labeled *ProviderB*. We cannot reveal the names of the proprietary databases since their terms of use forbid comparative benchmarking. They are among the most popular and accurate databases and they are both available to the public.

Figure 9.9 compares error distance across the four models. The X-axis represents the cumulative error distance, while the Y-axis shows how many points fall within that particular error distance band. For instance, the second column shows that *HigherAcc* places 80.5% of the predicted locations within 20 km of their actual location in the ground truth set. The figure shows that both of our instances significantly outperform the commercial databases. Table 9.1 also compares the values for median error, percentage of points with error of less than 10 km, and RMSE. Lastly, our results also surpass prior academic work, which had error in the order of hundreds of km. The disadvantage of our approach is that our *HigherCov* model still only has an IP coverage of 52.21%, while commercial databases cover more than 90% of IP space. However, our coverage still far surpasses prior academic work in the area. We discuss one approach to further improve coverage in the next section.

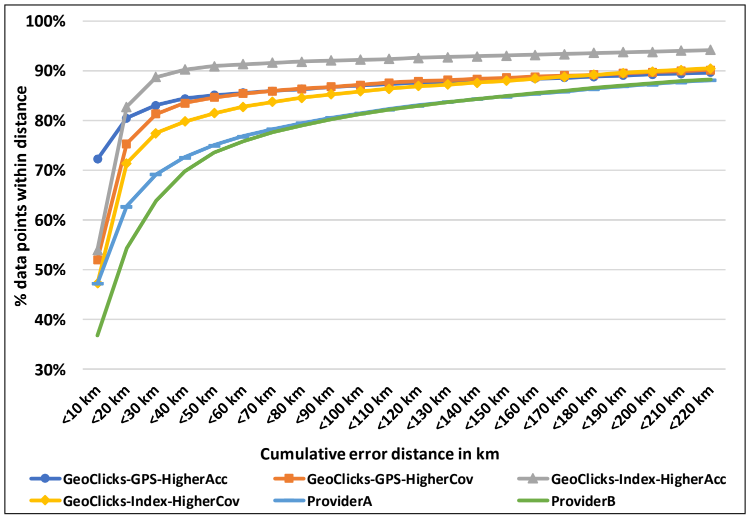


Figure 9.9: Error distance in kilometers with ten-fold cross-validation for our approaches and two state of the art commercial geolocation services.

Table 9.1: Comparison between three instances of our two approaches and two state of the art commercial geolocation databases, across multiple metrics.

	Median error	**% distance <10km**	**RMSE in km**
GeoClicks-GPS-HigherAcc	**4.5**	**72.2%**	893.4
GeoClicks-GPS-HigherCov	9.5	52.1%	711.1
GeoClicks-Index-HigherAcc	9.2	54.0%	1327.4
GeoClicks-Index-HigherCov	10.7	47.3%	1498.6
Commercial Provider A	11.1	47.2%	545.9
Commercial Provider B	16.7	36.7%	**545.3**

Finally, Table 9.2 demonstrates the effect of our proposal to weigh the centroid locations in the second step by the URL confidence scores computed in the first step. We obtain an improvement of 4.2% in the most important error band of error smaller than 10 km.

9.6.4 Academic Baseline

We now evaluate the two click based approaches against our previously published query logs approach [169], which we also described in Chapter 7 of this dissertation. To the best of our knowledge, this is the only other academic IP geolocation approach which uses search engines logs for IP geolocation.

Table 9.2: Improvement in accuracy when using weighted centroids for IP range locations in GeoClicks-HigherCov.

Cumulative Error in km	**Unweighted Centroids**	**Weighted Centroids**	**Improvement for weighted**
< 10 km	49.9%	52.1%	4.2%
< 20 km	74.7%	75.4%	0.9%
< 30 km	81.1%	81.3%	0.2%

To re-implement the query based approach we mined Bing logs over a period of 28 days, ending on October 26th, 2018. We reduced bias caused by single addresses by selecting one query instance per IP per day. Using the same methodology as described in Chapter 7, we then retained queries with local intent such as business searches, directions, local cinema showtimes, and local weather. Finally, we filtered the remaining impressions to keep only the ones that contained explicit locations. This resulted in 374 million queries that were issued from 3.4 million distinct /24 (256 IPs) buckets. After grouping and filtering locations by IP range, we evaluated the approach using the same ground truth of 70 million IP addresses.

Figure 9.10 shows that the query logs approach generally has lower accuracy than the click based approaches, with the exception of accuracy at <10 km, where the baseline surpasses our web index based variant that is tuned for higher coverage, but still comes up short when compared to our three other instances.

Table 9.3 contains a comparison across several metrics. Our four click based instances significantly outperform the query logs approach in RMSE and three of four variants outperform the baseline in median error and accuracy at the 10 kilometers threshold. In the last column we also introduce a fourth metric which combines accuracy and IP coverage. Studying this last metric yields an interesting conclusion. Whereas our high coverage web index based variant has lower accuracy than the query logs approach, it comes out on top if we also consider IP coverage. In fast, this variant surpasses both the baseline, and all the other variants, by a large margin.

In conclusion, our click based approaches outperform a baseline based on mining query logs, but the choice of using one click based variant over another

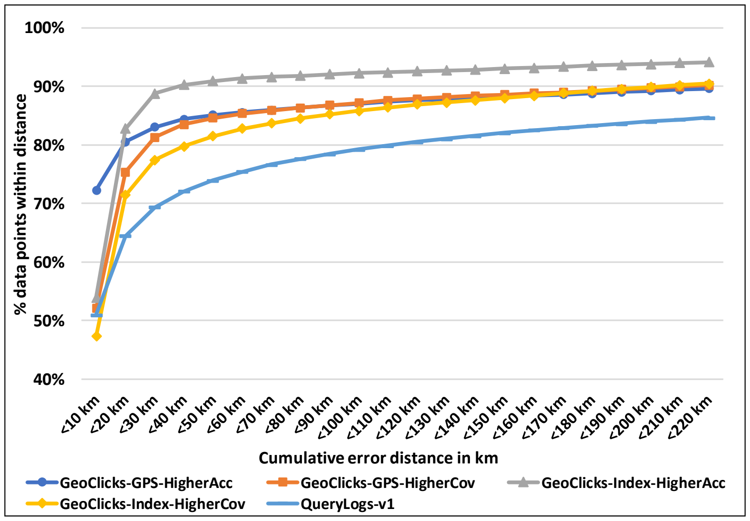


Figure 9.10: Error distance in km with ten-fold cross-validation between our approaches and a state of the art academic baseline based on mining query logs.

Table 9.3: Comparison between three instances of our two approaches and an academic baseline that uses search engine logs [169]. This academic baseline is also based on our work, as described in Chapter 7.

	Median error	% distance <10km	RMSE in km	% IP coverage w/ err <10km
GeoClicks-GPS-HigherAcc	**4.5**	**72.2%**	893.4	2.2%
GeoClicks-GPS-HigherCov	9.5	52.1%	**711.1**	28.7%
GeoClicks-Index-HigherAcc	9.2	54.0%	1327.4	10.9%
GeoClicks-Index-HigherCov	10.7	47.3%	1498.6	**35.7%**
Query Logs Geolocation [169]	9.6	51.0%	2126.4	53.5%

depends on the application. For applications in need of higher coverage, the index based approach is the best option. However, if instead the goal is to achieve the highest accuracy, then a GPS based approach is the best option.

9.7 Agreement and Overlap Between Approaches

Our two approaches are based on location information extracted from GPS sensors and the body of web pages, respectively. In this section we aim to quantify the degree to which there is overlap and agreement between these techniques. We compare the higher coverage variants. The variant based on

Table 9.4: Overlap between the GPS and web index techniques at the /24 IP range level.

	Common in both approaches	Only in GPS	Only in index
Covered /24 IP ranges	**821,571** (62.0% of GPS, 48.6% of index)	**504,109** (38.0% of GPS)	**870,971** (51.4% of index)

GPS data has a ground truth IP coverage of 55.2%, while the ones using data from the body of web pages has a higher coverage at 75.4%.

Table 9.4 contains a breakdown of the overlap and differences between methods, at the IP range level. The method based on GPS locations has a total coverage of 1.32 million IP ranges, while the one based on mining web content has a coverage of 1.69 million IP ranges. Their intersection results in 821,571 ranges. This result shows that while the techniques output many IP ranges in common, they are also quite complementary, with 504,109 IP ranges only covered by the GPS method, and 870,971 IP ranges only covered by the web index approach.

We now turn our attention to analyzing the IP ranges which the methods share in common. Figure 9.11 displays the distribution of distances between the locations found by the two approaches. We generated this graph by intersecting the IP ranges, and then calculating the distance between the locations found by the two approaches in each of the shared IP ranges. The result show that even though the approaches derive locations from very different data sources, they have good agreement as 74.5% of the distances are within 10 kilometers of each other.

Finally, an obvious question one may ask is if the intersection of the two approaches yields better ground truth results than the individual methods. First, we retain the common IP ranges where the two locations are within 20 kilometers of each other. Second, for each IP Range we take the mean point between the two locations. Third, we evaluate the resulting dataset against the ground truth. Figure 9.12 shows that taking the intersection of the approaches results in better accuracy across the entire distance spectrum. The median error is 8.7 kilometers, which makes it second in accuracy only to the higher accuracy variant of the GPS approach. The RMSE is only 375.5, which is the lowest across all variants and baselines. The combined approach

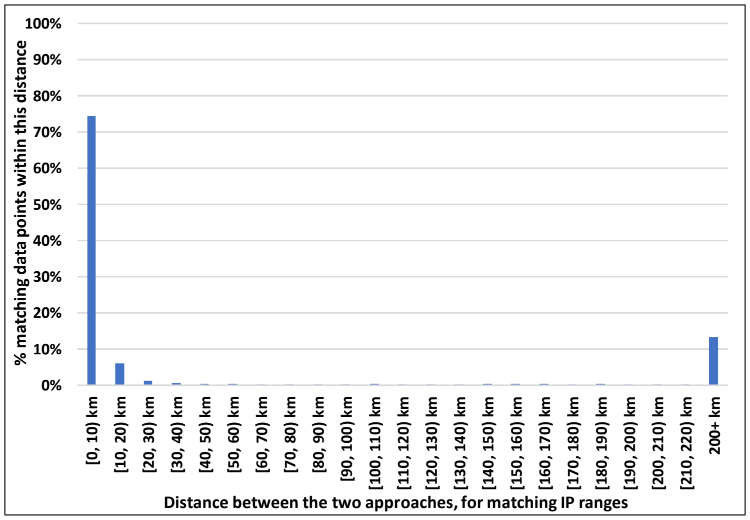


Figure 9.11: The agreement in distance for the IP ranges shared by both approaches. The first bar in the graph shows that whenever the two approaches find locations for the same IP range, 74.5% of the time the locations emitted by the two approaches are within 10 kilometers of each other. Please note that the agreement results do not use any ground truth information, they simply show how often the two approaches agree on the location of shared IP ranges.

has a ground truth IP coverage of only 40.5%, which as expected is lower than either of the two approaches.

9.8 Conclusions

In this chapter we studied propagating locations from IPs with known location to IPs with unknown location, using user clicks. Our research has practical applications in improving search engine personalization, as well as other online services. It can also augment academic research in geographic user cohort modeling. Results show that our two proposals significantly outperform two widely used commercial geolocation databases. The results also show that our two proposed approaches are complementary with roughly half of the IP ranges overlapping, and their intersection is highly accurate with a median error of only 8.7 kilometers.

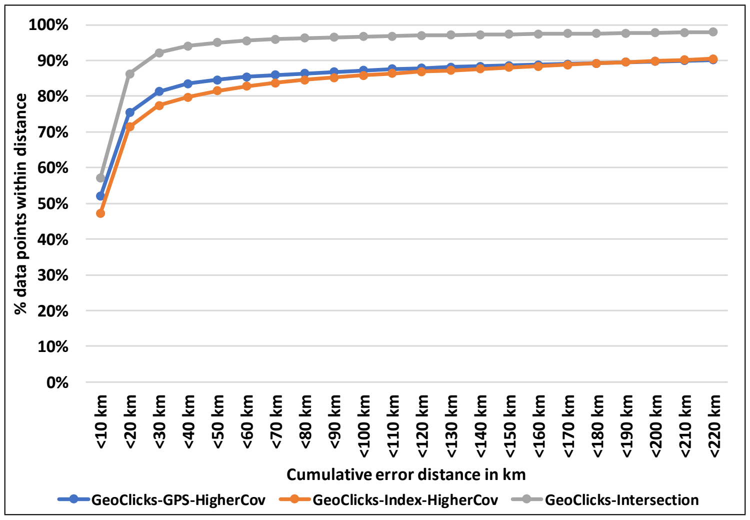


Figure 9.12: Evaluation against ground truth of the blended results obtained by intersecting the output of the GPS and index approaches at the IP range level, and retaining the instances where the two approaches agree on a location (within 20 kilometers). The intersected output yields better accuracy than any of the individual approaches.

Chapter 10

Revisiting Query Logs

In Chapter 7 we improved the accuracy of commercial IP geolocation databases using user queries which contain explicit locations. Here we propose two improvements to that preliminary approach by incorporating techniques from Chapter 9. First, we remove the dependency on reverse geocoding and we instead use density clustering to output pairs of coordinates instead of city names. Second, for each IP range we use the weighted centroid based on the underlying confidence of each data point.

10.1 Approach

We will now describe the processing steps in the revised approach in more detail:

1. **Extract queries** and corresponding IP addresses from query logs.
2. **Reduce bias** by retaining one query instance per IP per day.
3. **Filter impressions**, keeping the ones likely to contain locations.
4. **Extract locations** from the queries that remain.
5. **Perform clustering** on the raw extracted coordinates at the /24 IP range level.
6. **Adjust the resulting centroid** for each IP range by using the confidence of each underlying location extracted from queries.
7. **Test the modified geolocation database** against the ground truth.

We extracted impressions from 28 days of Bing logs, ending on October 26th, 2018. To reduce potential bias introduced by single IP addresses with many duplicate queries, we retained one query instance per IP per day, across each of the 28 sampled days. Next, to increase the chance that the queries contain explicit locations, we performed a filtering step where we selected the impressions with local intent, such as business searches, directions, local cinema showtimes, and local weather. This resulted in 374 million queries that were issued from $\approx$300 million IP addresses and 3.4 million distinct /24 ranges (256 IPs in size). Finally, we extracted locations from user queries as previously described in Chapter 7. Each extracted location consists of coordinates and a confidence value.

We will now describe the modifications we have made to the preliminary approach, and explain the motivations for making these changes. We again use a /24 (256 IPs) granularity for IP ranges. However, we do not perform reverse geocoding to convert the extracted coordinates to city boundaries and we do not count city ID occurrences in each IP range. Instead, we now directly use the raw coordinates and their confidence without a normalization step and we group the locations using density clustering. For the clustering step we followed the method and parameters we used to cluster locations from clicks in Chapter 9. The reason for this change is that reverse geocoding is costly, time consuming, and often imprecise. Forgoing reverse geocoding and using clustering instead also helps because nearby points can directionally help in locating a cluster, even if the points close to the outer edges are possibly in a different city. Removing this step also simplifies the approach, since instead of converting from queries, to coordinates, then to cities, we directly use the coordinates. We use the largest cluster from the output of the DBSCAN clustering algorithm. Finally, we adjusted the location of the centroid by using the confidence of the underlying coordinates as weights. The motivation for this change is that, as we demonstrated in Section 9.6.3 and Table 9.2, using a weighted centroid can alone increase accuracy by a few percentage points.

10.2 Evaluation

For evaluation we use a ground truth set which contains 70 million IP addresses with known location, compiled during the 28-day period ending on October 26th, 2018. The data was collected from Bing users who opted in to provide GPS location information along with their queries. We compare our approach against two state of the art commercial IP geolocation databases. We also evaluate it against the preliminary version from Chapter 7, and against the high coverage variants of the click based approaches from Chapter 9.

10.2.1 Commercial Baselines

Figure 10.1 compares cumulative error distance between our approach named *QueryLogs-v2*, and two commercial geolocation services named *ProviderA* and *Provider B*. Our improved query based method significantly outperforms the two commercial services. This large difference is maintained across the entire error distance range.

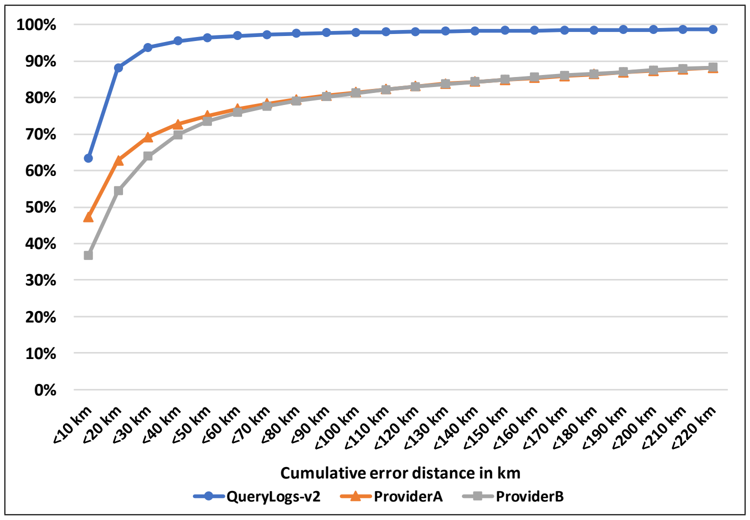


Figure 10.1: Error distance comparison between the improved query logs approach and two state of the art commercial IP geolocation databases

Table 10.1 evaluates our approach against the commercial baselines, across multiple metrics. Here our method again significantly outperforms the commercial databases. Our median error is lower at 7.6 kilometers than the baselines, which yield a higher error of 11.1 and 16.7 kilometers, respectively. The percentage of points that fall within 10 kilometers of their actual location is much higher at 63.3% when compared to the commercial services which attain only 47.2% and 36.7%, respectively. Finally, our RMSE value is lower and therefore better than both of the commercial alternatives.

Table 10.1: Comparison between our improved query based approach and two state of the art commercial geolocation services, across multiple metrics.

	Median error	**% distance <10km**	**RMSE in km**
QueryLogs-v2	**<u>7.6</u>**	**<u>63.3%</u>**	**<u>467.7</u>**
Commercial Provider A	11.1	47.2%	545.9
Commercial Provider B	16.7	36.7%	545.3

10.2.2 Academic Baselines

To get an idea of the performance of the enhanced query based approach in the context of other techniques presented in this dissertation, we compare it to the older preliminary version, and to the user click methods from Chapter 7.

Figure 10.2 displays the cumulative error distance curves for *QueryLogs-v1*, which is the previous version, *QueryLogs-v2*, which is the new enhanced version, and the two high coverage click based approaches. The graph shows that the updated query based version manages to outperform both the older approach, and all the click based approaches, by a large margin.

The metrics comparison in Table 10.2 similarly shows that the new approach is outperforming the baselines in median error, percentage of points within 10 kilometers of their actual location, and root-mean-square error. However, when we combine accuracy and coverage to show the percentage of ground truth IP coverage with error smaller than 10 kilometers, we can see that the click based web mining approach achieves a better result. The reason for this result is that even if the click based approach has lower accuracy, it

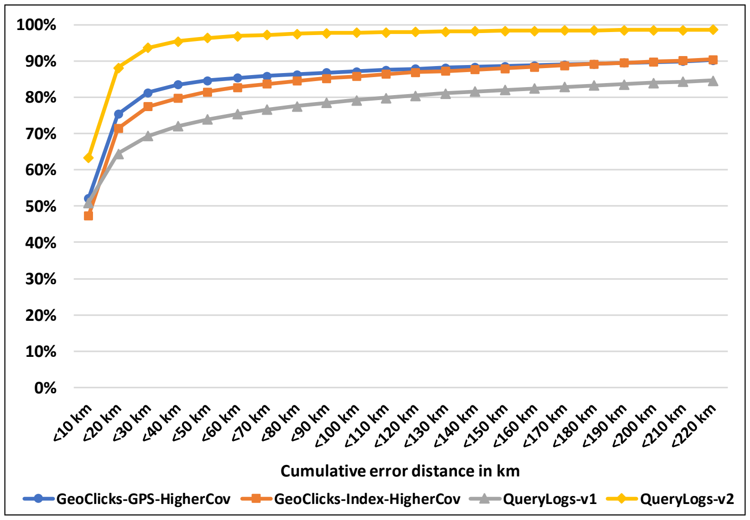


Figure 10.2: Comparison of error distance in kilometers between the old approach based on query logs, the new one, and the high coverage variants of the click based approaches from Chapter 9

Table 10.2: Comparison between the high coverage click based approaches, the preliminary query based approach, and the updated query based approach we presented in this chapter, across several metrics.

	Median error	**% distance <10km**	**RMSE in km**	**% IP coverage w/ err <10km**
GeoClicks-GPS-HigherCov	9.5	52.1%	711.1	28.7%
GeoClicks-Index-HigherCov	10.7	47.3%	1498.6	**35.7%**
QueryLogs-v1	9.6	51.0%	2126.4	27.3%
QueryLogs-v2	**7.6**	**63.3%**	**467.7**	28.6%

has higher ground truth IP coverage. The technique proposed in this chapter covers 31.4 million IPs from the ground truth set, while the winning click based approach covers 52.5 million.

In conclusion, the improved geolocation approach achieves significantly better results across all metrics when compared to the preliminary version from Chapter 7, and it also outperforms the click based approaches across most metrics.

Chapter 11

IP Location Interpolation

In this chapter we study the concept of *IP location interpolation*, which is a process to extrapolate the location of an entire IP range by using the locations of a few underlying IP addresses with known location. We first study the colocation of IP ranges, by computing the distance between pairs of addresses that are in the same IP range. Based on the preliminary findings, we then introduce an approach to perform IP interpolation at the entire IP range level. Finally, we evaluate our approach at multiple levels of granularity, using a large and geographically diverse ground truth set of 8.9 million IP addresses.

Previous research has used multiple terms for extrapolating the location of an entire IP range from a few individual addresses, including clustering [70, 71], geographic locality [97], block-based geolocation [59, 170], segment inference [30], IP segmenting [62], and aggregation [31]. Here we call this concept IP location interpolation.

Early work by Padmanabhan and Subramanian proposed a technique called *GeoCluster*, which consists of obtaining IP network prefixes from BGP router table dumps, and then propagating IP addresses with known location throughout these prefixes [70, 71]. They also proposed breaking up larger network blocks into smaller segments, if the contained ground truth IPs did not agree on a location. There are three issues with using network prefixes from BGP blocks. First, as the authors mention, BGP dumps only cover a fraction of all IP address space. Second, a BGP table is a summarized snapshot of the Internet, as viewed from the perspective of a single router, so it contains large

IP ranges. In order to get a more granular breakdown of IP ranges, one would need to extract tables from many different geographically dispersed routers, which is generally not feasible. Third, individual network blocks in BGP tables do not necessarily correspond to geographically co-located IP addresses. In fact, Padmanabhan and Subramanian show that their GeoCluster approach only achieves a median error of 685 kilometers on a ground truth set of 181,246 IPs in the United States. Alidade [31] makes an even stronger assumption that all of the IPs in a prefix must be located in the same location. However, Freedman et al. have studied the geographic locality of BGP network blocks and they have found that about half of the prefixes of size /8 to /15 contain IP addresses in multiple geographic locations, which further shows that using block sizes as defined by the BGP table may not always be the best option [97]. In this chapter we remove the dependency on incomplete BGP routing information and we evaluate interpolation across the entire IP address space covered by our large ground truth set.

Structon [30], proposed by Guo et al., uses interpolation as a technique to increase the IP coverage of a web mining based geolocation approach. They assume all IP addresses in the same /24 segment are in the same city. They iteratively apply majority voting to increase IP range sizes until they reach a netmask of size /18. They also combine interpolation with information from BGP routes and traceroutes. However, they do not present any interpolation specific experiments and they do not provide a motivation for performing interpolation in this specific way and at this specific granularity. Their approach is also focused exclusively on a few hundred cities in China. Here we instead systematically test different range and we also perform experiments worldwide.

Liu et al. also apply IP location interpolation as part of a location sharing social network based geolocation method called Checkin-Geo [62]. However, they directly use /24 segments without much explanation, they do not describe their interpolation approach, and like Guo et al. they only focus on IP addresses in China. Since our ground truth set covers most countries in the world, our experiments are more representative of conducting interpolation worldwide.

Finally, Lee et al. use a majority rule vote with a threshold of 80% to assign individual IP address locations to entire IP ranges of sizes /24, /25, and /26, respectively. Unfortunately, the paper does not evaluate the technique against ground truth data. Comparatively, we do evaluate interpolation against a large ground truth set of 8.9 million IP addresses.

In summary, our approach differs in significant ways from previous work. First, we separate the concept of IP interpolation from the usage of BGP prefixes by considering the entire address space, instead of using the subset of prefixes covered by the routing table. Second, we systematically evaluate multiple IP range sizes, to determine the impact that netmask size has on accuracy. Third, we use the largest ground truth set ever reported in IP local interpolation literature. Fourth, instead of focusing on a single region we test interpolation across the entire world.

11.1 Colocation of IP Range Addresses

We will now test the assumption that IP addresses that are in the same contiguous IP range are likely to also be located in the same geographic area. We evaluate this hypothesis using a large ground truth set of 8.9 million IP addresses. Our **ground truth set** contains IP addresses with known location, and was compiled during the 28-day period ending on December 1st, 2017. It is one of the largest and most diverse ground truth sets used in geolocation literature. It was derived from the query logs of the Bing search engine from devices with global positioning sensors, where users opted-in to provide location information. The dataset contains both mobile and fixed broadband IP addresses, since users often connect their mobile phones to their home Wi-Fi. It covers the entire world. Throughout this chapter we used this ground truth set for both training and testing by performing **ten-fold cross validation**, by randomly splitting the data into ten folds and repeatedly using 9 folds for training and the last fold for testing. We report the results as the average of all the runs. We never had access to the raw location data. Instead, the dataset was anonymized by an automated pipeline by aggregating all locations reported for an IP address, then adjusting the centroid of each IP address by 584 meters in a random direction. IP addresses with a large variance in reported locations were removed as outliers. These anonymized coordinates cannot be used to pinpoint individual addresses, but can locate an IP at a neighborhood level. While throughout this paper we refer to this location data as derived from *GPS* for succinctness, the dataset actually covers all global positioning systems, including *GPS*, *GLONASS*, *Galileo*, etc. [160].

We segmented the IPv4 address space into IP ranges of varying lengths for netmasks between */28* (16 IPs) and */20* (4,096 IPs). We then extracted all pairs of IP addresses which are part of the same IP range, and are also in the ground truth set. So, for example, IPs 50.121.73.3 and 50.121.73.47 would form a pair in IP ranges with netmask */26* (64 IPs) to */20* (4,096 IPs), but not for IP range *50.121.73.0/28*, which is only 16 IPs in size. Since we know the location of each IP in the ground truth set, we were then able to compute the geographic distance in each pair.

Figure 11.1 presents the results as cumulative distance curves. The X axis represents the distance between pair items and the Y axis shows how many pairs are within that distance. For instance, if we look at the first column (*<10 km*) for the IP range size of 1,024 IPs, we can observe that roughly 60% of pairs in this type of IP range are within 10 kilometers of each other. We can draw two conclusions from the graph. First, the size of the IP range is directly proportional to pair distance. As the range size increases, the distance between pairs in the range also increases. For ranges that are 256 IPs in size, 88% of pairs are within 30 kilometers of each other, but that percentage drops to 77.9% for IP ranges with 1,024 IPs. Second, even if these preliminary results are promising, the graph shows that there is room for improvement. We cannot assign the locations of the pairs to the entire IP range, since in some cases the locations are contradictory. We must therefore find a way to further filter these IP ranges to retain the ones where the location signal is consistent.

11.2 IP Range Interpolation

We now apply our finding that addresses in the same IP range are often colocated geographically to the problem of increasing ground truth IP coverage. We propose performing IP range location interpolation by finding blocks where all ground truth IPs contained in that range are in the same geographic region, and then assigning the center of all IP coordinates to the entire range. Figure

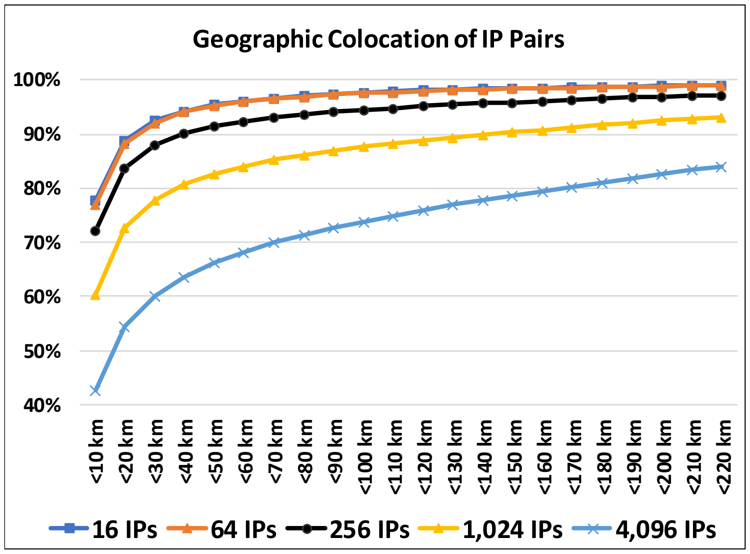


Figure 11.1: Cumulative distance between pairs of IPs in the same range, for different range sizes. For example, in the *<20 km* column, the value of the distance for IP ranges of *256 IPs* is 83.8%. This means that if we segment the IP space in contiguous ranges of size 256, a pair of IP addresses from the ground truth set that are in the same IP range are at a distance smaller than 20 kilometers from each other in 83.8% of cases.

11.2 shows an example where an IP range contains two ground truth IPs, both located in New York City. Since all the ground truth IPs contained in this IP range are located in the same region, we propose assigning the location New York City to the entire IP range.

To perform interpolation, for a given IP range size we first grouped all ground truth IPs by the given netmask. In each of these contiguous IP ranges we computed the pairwise distance between the ground truth IPs. We retained the IP ranges that contained at least n ground truth IPs in total, and *all* these IPs in the range are within m kilometers of each other. We then assigned the center of all these coordinates to be the location of the entire IP range.

We evaluated our proposal on multiple IP range sizes and multiple values of n and m using local parameter search on our ground truth set. We found that using a size of 256 IPs yields the best combination of accuracy and coverage. This IP range size is also often used by commercial IP geolocation databases.

We obtained good accuracy by setting the n parameter to 2. Setting it to 1 yields lower accuracy by 0.6 percentage points and higher coverage by

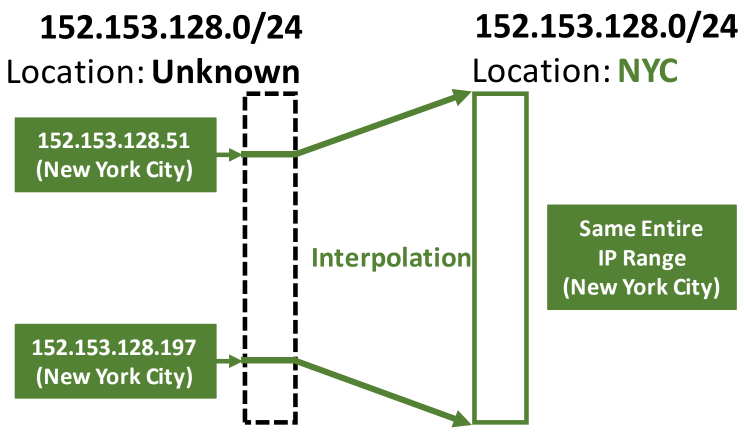

152.153.128.0/24
Location: Unknown
152.153.128.0/24
Location: NYC
152.153.128.51 (New York City)
152.153.128.197 (New York City)
Interpolation
Same Entire IP Range (New York City)

Figure 11.2: Example of IP range interpolation. Since IP range 152.153.128.0/24 contains two IP addresses with the same known location (coordinates in New York City), we propagate that location to the entire IP range.

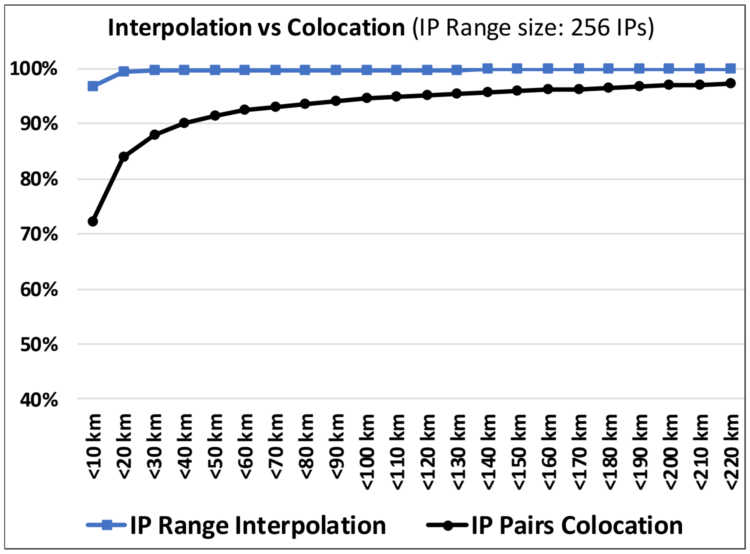


Figure 11.3: Comparison of IP range interpolation error distance to IP range colocation distance. Note that interpolation error distance represents the distance between the predicted location and the actual location as given by the ground truth set, while the colocation distance represents the pair-wise distance between IPs with known location.

0.8%, while setting it to 3 yields higher accuracy by 0.2 percentage points and a decrease in coverage by 0.8%. Finally, we set the m parameter to 25 kilometers. Setting it to 20 kilometers yields a 1 percentage point improvement in accuracy at 10 kilometers, at the cost of reduced coverage by 25%, while setting it at 30 kilometers results in an accuracy decrease of 1.5 percentage points but with an increase of 11% in coverage. Using these parameters (*n=2, m=25*), we filtered the 3.5 million distinct IP ranges of size 256 in the ground truth set down to 1.5 million ranges used for interpolation.

Figure 11.3 presents the evaluation result using ten-fold cross validation on the ground truth set. The interpolation curve shows the *error distance* between where our interpolation places the IP, and its actual location. Results show that 96.7% of IPs have a predicted location that is within 10 kilometers of their actual location, and 99.4% of them are within 20 kilometers. For comparison, we have also displayed the equivalent 256 IPs colocation curve from Figure 11.1, where IP range pairs were located within 10 kilometers for 72.3% of data points, and 20 kilometers for 83.8% of data points. In conclusion, if two or more IP addresses in the same IP range are located in the same geographic area, it is very likely that the rest of the IP range is also in the same area. Using IP range interpolation we increased our ground truth coverage from 8.9 million IP addresses to 382 million IP addresses. **In Chapter 12 we will further evaluate using interpolation to extend the IP coverage of a geolocation method based on traceroute paths.**

Chapter 12

Traceroute Location Propagation

In this chapter we use latency differences along the traceroute path, combined with interpolation at the IP range level, to improve IP geolocation. We base our approach on two assumptions. Our first assumption, which we have already evaluated in Chapter 11, is that addresses in a consecutive IP range are often located in the same geographic region. Our second assumption, which we study here, is that nodes which are close together in terms of latency on a traceroute path are also near in terms of geographic distance. We evaluate to what degree these assumptions are true using a large ground truth set of 8.9 million IP addresses.

In our preliminary investigation we define the concept of **latency neighbors**. We show that there is a direct relationship between latency differences along the traceroute path and physical distance in kilometers. We propose to exploit this property to improve IP geolocation. We then propose combining the concepts of traceroute latency neighbors and IP range location interpolation to improve IP geolocation. We interpolate locations from the training set to increase its coverage. Then, we use latency neighbors to further **propagate locations** from IP ranges with known locations to IP ranges with unknown locations, using traceroute latency neighbors in aggregate. We evaluate our approach against two state of the art commercial IP geolocation databases, using a large traceroute dataset and a large ground truth set. We show that our approach significantly outperforms two commercial geolocation databases.

12.1 Datasets

The public **traceroute dataset** contains 9 billion traceroutes collected between January and November 2017. We derived it from the *IPv4 Routed /24 Topology Dataset* [125] provided by the Center for Applied Internet Data Analysis (CAIDA). They collect this data through the Archipelago (Ark) Measurement Infrastructure, which spans approximately 208 servers located in 63 countries. Every 48 hours a random IP address is chosen in *each* /24 prefix, then the chosen IP addresses are individually probed by random Ark servers. Therefore, both the IP chosen per range and the Ark machine probing that IP change in time. While this allows for more data variety, it also prevents using the dataset for typical latency multilateration [39]. We further parse and apply post-processing on this dataset to extract latency neighbors, as described in Section 12.2.

Our proprietary **ground truth set** contains 8.9 million IP addresses with known location, compiled during the 28-day period ending on December 1st, 2017. We use the same dataset as described in Section 11.1. Please refer to that section for a detailed description of the dataset. Although we evaluate our approach on IPv4, methods described here can be equally applied to IPv6 IPs.

To aid in reproducing our experiments, we also perform our final evaluation with a second public training set extracted from **PeeringDB**, which is a self-reported database of worldwide peering points [171]. The dataset contains approximately 400 IP ranges spanning 128,000 IP addresses, along with geographic coordinates. Whereas the first ground truth set is proprietary and it mainly contains end user client IPs, this second dataset contains IP ranges that are part of the Internet peering infrastructure. To obtain it, we enumerate the Internet Exchanges in the database, then for each exchange we enumerate the facilities. We retain only the facilities which contain exact location coordinates. Since IP ranges are published at the Internet Exchange level, we determine a location consensus among the coordinates of all the facilities belonging to an exchange. If all facilities are located in the same city, then we output the IP range and the consensus coordinates. Along with this

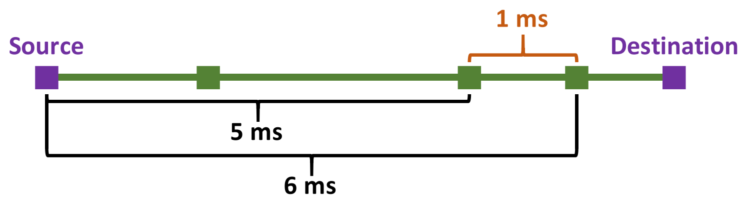


Figure 12.1: Example of latency neighbors for $X <= 1$ ms. Since the latency difference between nodes 4 and 5 on the traceroute path is 1 milliseconds, we consider them latency neighbors.

dissertation we are making the traceroute dataset parsing library and the PeeringDB parsing and generation library available as **open source**. We are also publishing a snapshot of the PeeringDB dataset.

We carried out our experiments through January 2019. The reason our traceroute and IP location ground truth data sets are from 2017 is that the Department of Homeland Security, which now hosts the dataset, restricts access to data newer than 12 months [172].

12.2 Latency Neighbors

We define *latency neighbors* as pairs of nodes along a traceroute path that are within X milliseconds from each other. We obtain the latency difference by subtracting the round-trip times between the source IP and the two neighbor candidates. Figure 12.1 shows an example of two nodes along a traceroute path which are at 5 and 6 milliseconds distance respectively from the source IP, which results in a latency difference of 1 millisecond. Since a pair can appear in multiple traceroutes, we aggregate all such instances and make a decision on the median round trip time. This aggregation has an added benefit of removing outlier pairs where the latency across multiple readings is too variable and the median becomes too high. To be considered latency neighbors, the nodes do not have to be located consecutively on the traceroute path, as long as they are within X milliseconds of each other.

Our assumption is that traceroute neighbors that are close together in terms of latency are also close geographically. As a preliminary test of this assumption, we extracted all latency neighbors from the traceroute dataset that were at most 10 milliseconds apart. Note that all latencies are from round-trip measurements, so the actual latency between them was at most 5 milliseconds. Previous research has found that packets travel in real networks at about 4/9 the speed of light, or about 133 kilometers per millisecond [86]. More recent research has suggested the speed in practice is even lower at 62.7 kilometers per millisecond on average [80]. We then further filtered these neighbor pairs to retain only the ones where both IPs were also present in our ground truth set with known IP locations. Since we required both exact neighbor IPs to be present in the ground truth set, this resulted in only 2,000 pairs, which is an extremely small coverage that makes it difficult to draw overall conclusions. The results show 65% of the neighbors are within 10 kilometers of each other. Although the results are promising, there is still a need to explore ways to increase ground truth coverage and to develop a systematic way to propagate locations over traceroute paths.

12.3 Location Propagation Evaluation

We perform location propagation by combining the concepts of latency neighbors and IP interpolation. First, we interpolate the ground truth set to increase its coverage. Then, we propagate these locations from IPs with known location to IPs with unknown location, through latency neighbors. We use the ground truth set both for training and testing by using ten-fold cross validation, where we propagate locations using 9 folds and we test using the last one. We report the results as the average of all the runs. The balance of accuracy and coverage of the model can be adjusted by varying two parameters used in determining latency neighbors: X, the maximum latency difference between two nodes and Y, a new parameter which restricts the maximum RTT between the source IP, and any of the two neighbors. In Figure 12.1 the maximum latency difference between the source IP and any of the two candidate latency neighbors is 6 milliseconds. If we set Y to be 6 or larger, then the two highlighted neighbors would be extracted as valid.

To find the optimal values we performed a local parameter search. We found that as we increased the maximum latency difference X from 1 to 4 the accuracy at *<10 km* decreased and the coverage increased. The same

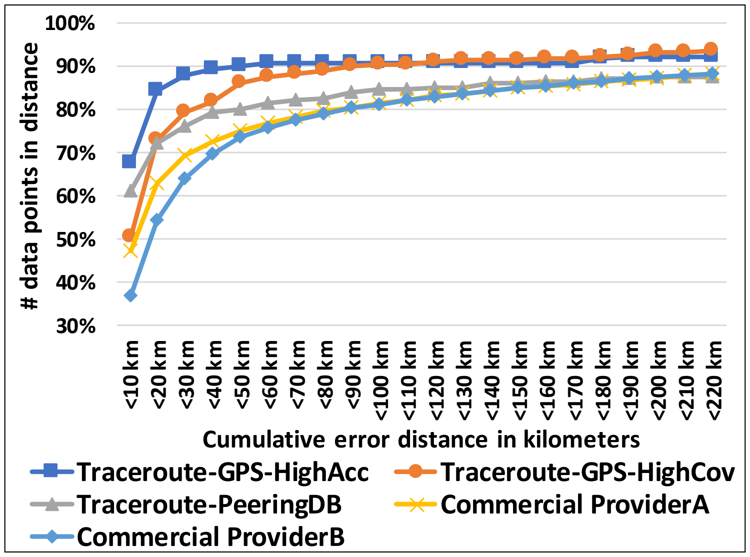


Figure 12.2: Cumulative error distance results that compare three instances of our approach to two state-of-the-art commercial IP geolocation baselines.

was true for varying the maximum RTT parameter from 1 to 10. The curves *Traceroute-GPS-HighAcc* and *Traceroute-GPS-HighCov* in Figure 12.2 present two instances of these parameters that graphically demonstrate the effect on accuracy. The former variation plots the results for parameters $X{=}2$, $Y{=}2$ and the latter variation uses $X{=}3$, $Y{=}9$. The higher accuracy version has a coverage of 1.4 million IP addresses across 7,400 IP ranges with propagated location, while the higher coverage version has a coverage of 15 million IPs across 83,000 IP ranges. These IPs had a previously unknown location that we now determined using location propagation. This evaluation uses the interpolated GPS-based ground truth as both training and test set. We also ran the same experiments on the non-interpolated ground truth set and obtained very similar results, but at lower IP coverage. Figure 12.2 also shows the results for a third variation of our approach that uses the PeeringDB dataset with $X{=}3$, $Y{=}9$ for location propagation. Here the overall results roughly fall between the first two instances.

We compare these three variations against two state of the art commercial databases, one labeled *ProviderA* and the other labeled *ProviderB*. We cannot reveal the names of the proprietary databases since their terms of use forbid comparative benchmarking. The results show that all three variants of our

Table 12.1: Comparison between three instances of our two approaches and two state of the art commercial geolocation databases, across several metrics.

	Median error	**% Err <10km**	**RMSE in km**
Traceroute-GPS-HighAcc	**4.3 km**	**67.7%**	**329.3**
Traceroute-GPS-HighCov	10.1 km	50.5%	423.6
Traceroute-PeeringDB	8.4 km	61.1%	2124.9
Commercial Provider A	11.1 km	47.2%	545.9
Commercial Provider B	16.7 km	36.7%	545.3

approach consistently outperform the commercial databases in error distance. Table 12.1 also compares our variants to these two baselines across multiple metrics. Our three instances outperform the commercial databases both in terms of median error (lower is better) and percentage of data points with error <10 km (higher is better). The last column of the table displays root-mean-square error, which is a metric more heavily influenced by outliers. It shows that the two instances trained on the proprietary dataset have a better (lower) RMSE than the commercial providers. However, it shows that the instance derived from PeeringDB data contains some outliers with high error distance. One potential explanation for these outliers is that sometimes PeeringDB IP ranges contain IPs that interconnect datacenters that are far from each other, so errors caused by these IP ranges result in large error distances.

12.4 Conclusions

We investigated and combined two IP geolocation approaches, one based on IP range interpolation, and the other one based on location propagation over traceroute paths. Our combined technique significantly outperforms state of the art commercial databases by up to 31 percentage points at error distance smaller than 10 kilometers. To aid in reproducing our results, we are making the traceroute dataset parsing library, and the PeeringDB parsing and generation library available as open source. We are also publishing a snapshot of the PeeringDB dataset.

Chapter 13

Extracting Location Information from Bulk WHOIS Databases

To avoid conflicts, public Internet IP addresses are unique. Early on, the allocation of IP addresses to organizations was carried out by Jon Postel, who maintained the list of ranges and organization details by hand in a paper notebook [173]. Once the list grew too big, this role was formalized and taken over in 1988 by an organization named the Internet Assigned Numbers Authority (IANA). As the Internet kept on growing, in 1992 the Internet Engineering Task Force (IETF) recommended creating subsidiary organizations to manage allocations at a more regional level [174, 175]. Today, there are five of these Regional Internet Registries (RIRs), each responsible for managing the allocation of IPs at a roughly continent level [176]:

- **ARIN** stands for the American Registry for Internet Numbers and serves Antarctica, Canada, parts of the Caribbean, and the United States.
- **AFRINIC**, the African Network Information Center, serves Africa.
- **APNIC**, which stands for the Asia-Pacific Network Information Centre, serves East Asia, Oceania, South Asia, and Southeast Asia.
- **LACNIC**, which is short for Latin America and Caribbean Network Information Centre, serves most of the Caribbean and all of Latin America.
- **RIPE NCC**, which stands for Réseaux IP Européens Network Coordination Centre, serves Europe, Central Asia, Russia, and West Asia.

IANA assigns large IP blocks to the five RIRs, which in turn allocate smaller blocks to large organizations such as Internet backbone providers or Internet Service Providers. These smaller organizations can then further suballocate progressively smaller chunks of the IP address space to businesses or even individuals.

When a business suballocates a block to a customer, they are generally required to report the information of the customer to the corresponding Regional Registry. There is however one exception. ARIN, which is the Regional Internet Registry responsible for North America, gives businesses two options to report block assignments. The first option is to report the allocation back to ARIN. The second option is to maintain a separate decentralized RWhois server [177]. This server needs to be accessible on the public Internet and must return up-to-date reassignment information [178].

Each IP block record in WHOIS databases contains information about the organization which received the block, often including postal addresses. Figure 13.1 contains an example of WHOIS output for IP 128.180.1.16, which is allocated to Lehigh University. The requested IP address matches IP range `128.180.0.0/16`, so the WHOIS command returns information about that entire block.

Since WHOIS databases can be downloaded in their entirety, they can potentially be a source of IP geolocation with relatively high coverage. As shown in Chapter 5, there are a handful of papers that have previously used WHOIS information for IP geolocation [29, 31, 115, 116, 124]. However, they have various limitations. Some of them are missing an evaluation section [115], some use basic parsing that only supports US zip codes or single countries [29, 116], and others lack descriptions of the approaches taken to extract information [31, 116, 124]. To the best of our knowledge, no previous work has used multiple complete WHOIS databases. Instead, researchers have typically performed live WHOIS queries for a limited of IPs that were part of their ground truth. In contrast, we target the entire IPv4 space by parsing complete WHOIS databases. Also, there is no previous work which exclusively focuses and evaluates using WHOIS databases for geolocation in isolation from other approaches.

In this chapter we systematically study using WHOIS data for IP geolocation. To use this information, we had to overcome several technical challenges:

```
whois 128.180.1.16

NetRange:       128.180.0.0 - 128.180.255.255
CIDR:           128.180.0.0/16
NetName:        LEHIGH
NetHandle:      NET-128-180-0-0-1
Parent:         NET128 (NET-128-0-0-0-0)
NetType:        Direct Allocation
OriginAS:
Organization:   Lehigh University (LEHIGH)
RegDate:        1987-06-17
Updated:        2017-08-04
Ref:            https://rdap.arin.net/registry/ip/128.180.0.0


OrgName:        Lehigh University
OrgId:          LEHIGH
Address:        183 Computing Center, Building 8B
City:           Bethlehem
StateProv:      PA
PostalCode:     18015
Country:        US
RegDate:        1987-06-17
Updated:        2017-08-07
Ref:            https://rdap.arin.net/registry/entity/LEHIGH
```

Figure 13.1: Example of WHOIS record for IP 128.180.1.16, which is allocated to Lehigh University and corresponds to the larger block 128.180.0.0/16.

1. **Access to WHOIS databases is not standardized**. Each RIR has a different application process, and some only receive applications through physical mail. Once we were granted permission to access the databases we also also noticed that they require different connection methods (HTTP, FTP) and that the files are compressed in various archive formats.
2. **Data schemas are heterogeneous**. Field names and values are also not standardized across the databases. A manual verification reveals that while ARIN records for North America contain separate fields for city, state, and country, other RIRs such as APNIC in Asia and AFRINIC in Africa combine postal addresses in a single `address` field. Even worse, RIPE NCC in Europe entirely strips any location information from their records due to privacy concerns. In fact, IETF used to have a working group tasked exclusively with standardizing WHOIS information. In their 2013 meeting they held a presentation on the differences between RIR data schemas, where they specifically highlighted location information as not being standardized [179]. Also, accessing secondary RWhois servers further introduces schema variability, with potentially hundreds of variations.
3. **Data quality is variable**. The quality of data varies from database to

database and even from record to record. Some records are missing fields, and others contain placeholder values.

4. **Collecting RWhois data requires active crawling**. While WHOIS databases are mantained in centralized locations and therefore can be downloaded in bulk, the Referral Whois system allowed by ARIN in North America is decentralized. Therefore, we need to crawl this information from disparate servers that run a variety of WHOIS software.
5. **Location normalization**. Depending on the database, postal addresses are either stored at the IP block level, or they are stored in the separate records of the asignee organization. Furthermore, since locations are inserted manually into WHOIS records, we need to standardize and obtain coordinates for each address.
6. **Overlapping and duplicate IP ranges**. Entire network blocks can get reallocated from one organization to another. Or they can be split into smaller ranges as they get assigned to smaller organizations. Each such operation results in a record in WHOIS databases. This means we can often encounter duplicate IP ranges, or smaller IP ranges which have a corresponding larger "parent" range. We need to perform merging and deduplication in order to not use IP ranges multiple times.

In the remainder of this chapter we first describe our approach to crawling, downloading, and parsing WHOIS records. We separately focus on collecting RWhois records, which to our knowledge has not been attempted before in related work. Then, we perform a preliminary investigation of the location data stored in these records. After we split, combine, and deduplicate network blocks, we complete a final evaluation.

13.1 Collecting, Parsing, and Normalizing Records

At the onset of our study we obtained and manually examined the bulk WHOIS databases from the five Regional Internet Registries. To overcome the aforementioned differences between databases, we implemented a library that facilitates downloading, unpacking, and parsing WHOIS records across the five

Organization name: Org-Name, Organization-Name, Customer Organization, Org-Name;I, OrgName
IP Range: CIDR, IP-Network, IP-Network-Block, Netblock, IP-Range, Network-Block, NetRange, inetnum, inet6num
Address: Address, Org-Address, Customer Address, customer-address, Address-1
City: City, Org-City, Organization-City, Customer City
State: Org-State, State, State-Province, StateProv, Organization-State, Customer State/Province, State/Prov
Postal Code: Postal-Code, Org-Zip, Customer Postal Code, Organization-Zip, PostalCode
Country: Country, Country-Code, Org-Country, Organization-Country, Customer Country Code

Figure 13.2: Examples of normalizing field names across the five RIR WHOIS databases

```
0, 00000, 99999, private, private residence, P.R., Private Address, Private Addr,
Private Addr., Private Resident, Private Res., PRIVATE CUSTOMER, private-address,
Unavailable Street, 1 Unavailable Street, Unavailable St, Unavailable Str,
Unavailable St., Unavailable Str., Unavailable Address, Unavailable, Street Not
Available, Address Not Available, Street N/A, Address N/A, N/A Street, N/A Addr,
N/A Addr., N/A Address, N/A St, N/A St., N/A Str, N/A Str., N/A, n.a., n.a, 1
na, Unknown, Postal Address, No info, Not Defined, Undefined, null, _None, None,
Address, Country, City, Postal Code, Street, PostalCode, Private Data, SERVER, fake
st, fake st., FakePostalCode, Fake, FakeTown, Fakeville, FakeSuburb, Fake_State,
Fake City, Fakeplace
```

Figure 13.3: Examples of invalid yet commonly occurring values in WHOIS records

registries. We have already made this WhoisParsers[1] library open source. Given authentication information, the library downloads each of the databases, unpacks them, and then parses them. The output of the library contains IP ranges, along with normalized addresses.

To normalize field names between the registries, we first parsed and counted the occurrence of each type of field. We then manually mapped the names to each other, as can be seen in the examples of Figure 13.2. To remove invalid placeholder values, we counted the occurrences of all values across all location related fields, and then we manually inspected the top results to compile a list of invalid values. Figure 13.3 shows a sample of these values.

We standardized locations in two steps. First, we located the postal addresses for each network block by joining IP range records with organization records. Depending on the database, the locations were either stored at the organization level, or at the IP block level. Second, to normalize the addresses we used the same approach as in Chapter 7, where we parsed locations using the Bing reverse geocoding API [69].

[1]`https://github.com/Microsoft/WhoisParsers`

Table 13.1: Statistics of information extracted from bulk WHOIS databases from the five Regional Internet Registries. The extra *RWhois* column is obtained by querying RWhois servers listed in the ARIN database.

	ARIN	**RWhois**	**AFRINIC**	**APNIC**	**LACNIC**	**RIPE**
Total WHOIS Records	8.1M	397K	230K	2.1M	435K	6.6M
↪ Organizations	3.6M	138K	2.4K	9.3K	9.1K	125K
↪ IPv6 Network Ranges	144K	N/A	30.9K	70.2K	17.3K	1M
↪ IPv4 Network Ranges	3.27M	259K	115.2K	1.0M	399K	4.1M
↪ With raw location candidate	3.26M	212K	6.7K	1.0M	399K	
↪ With extracted location	3.25M	212K	6.7K	622K	390K	
↪ After range dedupe	3.25M	210K	6.7K	622K	330K	

RWhois, which stands for Referral Whois, requires crawling information from many different disparate WHOIS servers. To crawl these servers, we first parsed the ARIN database to find networks that contain IP ranges with the field `ReferralServer`. Starting from the given IP range, we then crawled the entire range by repeatedly incrementing the starting IP address by 16 IPs.

13.2 Preliminary Investigation

We downloaded and parsed the five RIR databases in December 2018. Starting from the ARIN database, we also crawled Referral Whois servers for 336 networks, of which 249 returned valid WHOIS records. Table 13.1 displays statistics on the collected data. ARIN has the most number of organizations in the database at 3.6 million, with the rest of the RIRs each having 137,588 or less. In terms of addresses, RIPE has the highest raw number of IPv4 ranges (regardless of size) at 4.1 million, followed by ARIN with 3.27 million. We use locations extracted either at the organization level, or directly at the IP range level, depending on the record fields.

After extracting and normalizing raw locations, and after deduplicating ranges by picking the oldest record for an IP range, ARIN is left with 3.25 million locations, followed by APNIC at 621,966. From the table we can notice the location coverage varies among the databases. AFRINIC has only 6,735 raw locations after parsing 115,200 IPv4 network ranges. The reason for this significant difference is that most AFRINIC records do not contain location fields. Also, after normalization the number of APNIC locations drops from 1 million to 621,966. The reason is that in the APNIC database the locations are sometimes recorded in the `record` field, which is noisy and can contain other unrelated information. While we show a column for RIPE, as we previously discussed this database does not contain location fields due to privacy concerns. In conclusion, the amount and quality of location information varies considerably across the databases.

We now turn to studying the granularity of networks blocks stored in these databases. Figure 13.4 displays the popularity of each network block size in WHOIS network records. To allow relative comparison between databases, the results are shown as percentages. Note that here we count the number of records with a certain size, we do not count the total number of IPs in those records. We display IP range popularity for multiples of two, from 2^0 to 2^{10}. The *Other* column mostly contains network blocks of larger size, but there are a few exceptions of IP ranges that have a number of IPs that falls in between multiples of two but are also smaller than 2^{10}. The figure shows, for example, that 23.2% of network records in the AFRINIC database have a size of 256 IPs.

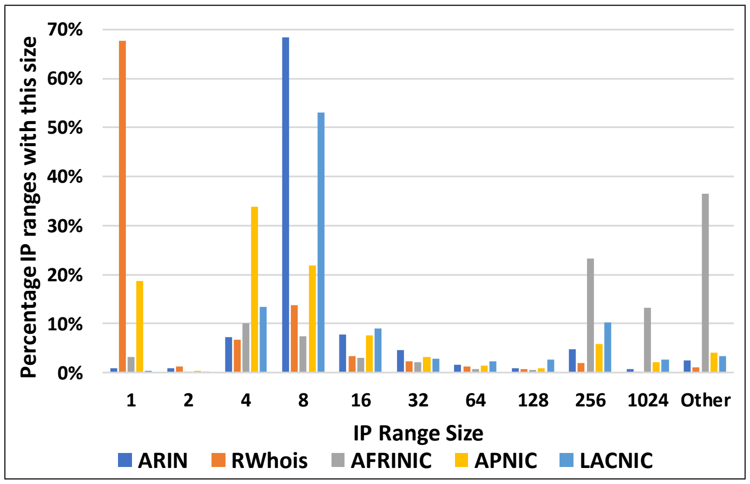


Figure 13.4: The granularity of IP Range sizes, across 4 of the bulk databases, as well as the crawled RWhois database.

The results in the figure lead us to several observations. First, 67.7% of the network records in the crawled Referral Whois database contain a single IP address. The reason for this large number of single IPs is that Referral Whois information is the most granular dataset since it is obtained directly from Internet providers which allocate IPs to small businesses. Second, the AFRINIC database has the largest percentage of records that fall in the *Other* category, at 36.5%. A manual inspection of the data behind the *Other* category reveals that 28.2 percentage points contain IP ranges between (1,024-65,536], with the rest of 8.3 percentage points contain network blocks that are larger in size. Third, blocks of 8 IPs are overwhelmingly popular in the ARIN and LACNIC databases, at 68.4% and 53.0%, respectively. The popular records are more spread out in the APNIC database, with record sizes of 4, 8, and 1 IPs being the most popular, in this order. In conclusion, the figure demonstrates that there are significant differences in IP range size popularity among the databases.

To get a sense of the IP coverage per network block size, in Figure 13.5 we display the total number of IPs in each IP range size, across all the databases. To obtain these numbers we simply multiplied the IP range sizes by the number of records that have that size. In contrast with the previous figure that only showed raw record counts, when we also take into consideration IP coverage it is apparent that IP ranges of size 1 and 2 have very small coverage, at 363,777 and 74,492 IPs, respectively. Furthermore, blocks of 256 IPs have the highest coverage among the sizes shown in the figure, followed by size 1,024 and size 8. If we were to continue the figure and display larger block sizes, those would overshadow these smaller IP range sizes in coverage. However, their location accuracy might be poor due to their large size.

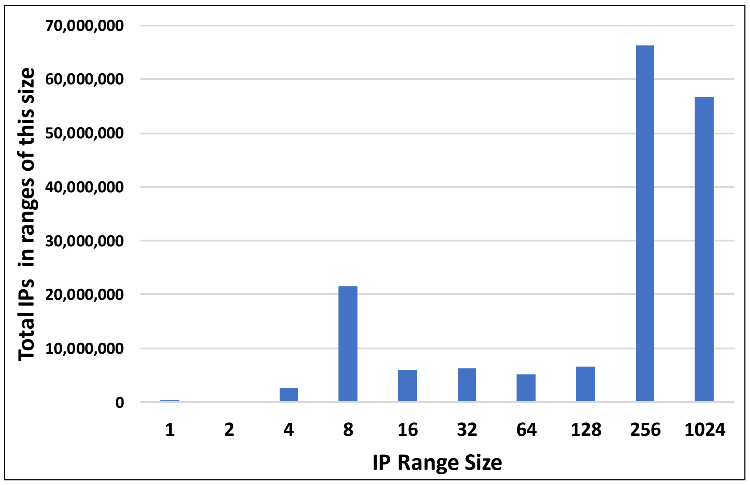


Figure 13.5: Total IP counts for each IP range size. We computed these numbers by multiplying the block size by how many blocks have this size.

As a preliminary evaluation, we determined the location accuracy of each database, for several IP range sizes. We first normalized locations and converted them to coordinates, as previously described. As shown in Table 13.1, some records did not contain valid locations and therefore this process changed the distribution of IP ranges. The AFRINIC data was the most impacted, since most AFRINIC records do not contain locations. Then, we performed deduplication on the records by finding exact duplicate ranges and retaining the record that was updated more recently. While this results in unique network blocks, they can and often still are overlapping. For instance, when a RIR allocates a large block to an ISP and then the ISP further re-allocates smaller subsets of the range, all of these records will overlap and show up separately in the databases. Using a ground truth set of ≈70 million IP addresses, we then determined the error distance for each database and block size.

Figure 13.6 displays accuracy for each database at 10 kilometers, across multiple range sizes. The bars show what percentage of ground truth IPs that fall in that database and range size combination are within 10 kilometers of the locations extracted from the database. The data reveals that accuracy varies between databases and within databases, depending on IP range size. Smaller IP ranges generally yield higher accuracy, across all databases. As the network block size increases, the accuracy drops. In some cases, such as ARIN, RWhois and LACNIC, this drop is significant, while in other cases such as AFRINIC and APNIC, it is more modest as these databases start out with a lower accuracy. This result suggests that larger network blocks generally cover a wider geographical region than smaller ones. The graph also shows that for smaller IP range sizes up to 64 IPs in size, ARIN and RWhois are much more accurate than the other databases. This suggests location data is higher quality in some databases when compared to others. The reason for the difference in accuracy is that the location data in ARIN and RWhois is stored in separate structured fields for City, State, etc., while the data in the other databases such as APNIC and AFRINIC is either stored in a single address record, or it is combined with other textual information which increases the amount of noise. Finally, it is interesting to note that the graph shows spikes of higher accuracy for certain network block sizes where their immediate neighbors have lower accuracy.

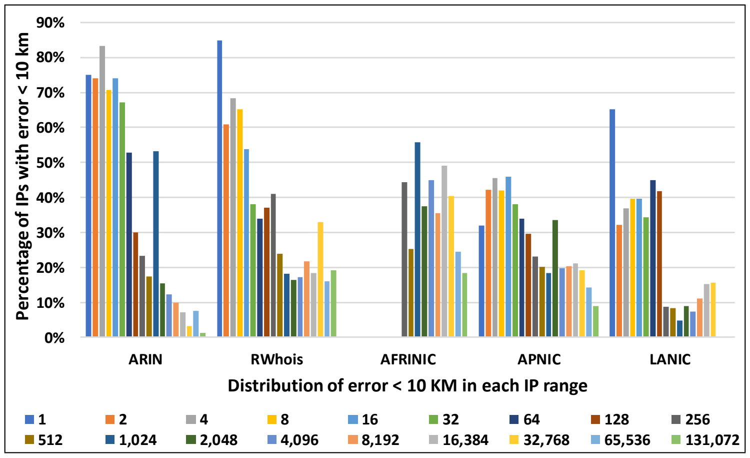


Figure 13.6: Accuracy per database and IP range at 10 kilometers. The X axis displays IP range sizes per database. The Y axis represents the accuracy at 10 kilometers, that is the percentage of data points where the real location in the ground truth set is within 10 kilometers of the location extracted from WHOIS records.

13.3 Segmenting and Combining Records

We performed the preliminary evaluation on the raw network block records, which often overlap either entirely or partially. These raw blocks are also variable in size, with the smallest ones containing a single address, and the larger ones containing more than a million addresses. To obtain a final dataset with non-overlapping blocks, we need to combine and merge these records. We also need to resolve any location disagreements that might occur. We propose the following approach:

1. **Parse the raw WHOIS records**, mapping similar fields between databases to each other. For example, the *Created* field in LACNIC is equivalent to the *RegDate* field in RWhois.

2. **Normalize locations** using the Bing reverse geocoding API [69]. Discard records which either do not have a location, or the location is not at city level granularity. For instance, if the granularity of the extracted location is at the county, state, or country level, we discard that record. We made this choice because locations that cover a large geographic area are not precise enough for IP geolocation.

3. **Remove exact duplicates** of IP ranges by grouping by IP range and retaining the record which was last updated. This step is necessary for cases when a WHOIS database contains the entire history allocations and reallocations for an IP range.
4. **Split ranges by given block size**. To allow combining network blocks of different sizes, we first pick a target block size. Then, we segment the raw WHOIS records by the target size. If a record has a size that is smaller than or equal to the block size, we do not modify it. If a record has a size that is larger than the target block size, we split it into multiple blocks, to match the target size. For example, if a raw record contains an IP range of 1,024 IPs, and the target block size is 256, we would split that raw record into 4 ranges of 256 each, that together cover the entire size of 1,024 addresses.
5. **Apply deduplication again on the target block size**. As before, apply deduplication for each distinct IP range by retaining the record with the latest update time. The difference compared to the first deduplication step is that in the first case we acted upon raw network blocks, while here we use the ranges that were split into a target block size.
6. **Determine consensus locations** for each distinct IP range. If a network block has a single location candidate, then we preserve that location. If the range has more than one location, we apply a technique inspired by Chapter 11, where we compute the pairwise distance between each data point in the range. If all pairs are within 25 kilometers of each other, we assign the centroid of the locations to the network block. If any pair is outside this distance, we discard the entire block. The reason for discarding blocks with conflicting locations is that it is possible that it actually contains smaller IP ranges not recorded in the database, which are actually in different locations. To avoid this problem, we decided to remove the entire block from the output.

Figure 13.7 shows an example of our approach. Both the left and right sides start from the same IP blocks in the first illustration, where the locations A and B have already been normalized. To segment the ranges, in the second illustration we choose a target block size. For the left side we picked a target block size of 64, while for the right size we picked a size of 128. If a raw network block, such as the leftmost one in the figure, is larger than the target block, we segment it into smaller ranges. After grouping the location data points by the target block size, in the third illustration we determine a location consensus, by retaining the IP ranges which have a consistent location, regardless of the number of underlying data points. An interesting observation is that, as can be seen in the example, picking a larger target range size does not necessarily result in higher total IP coverage.

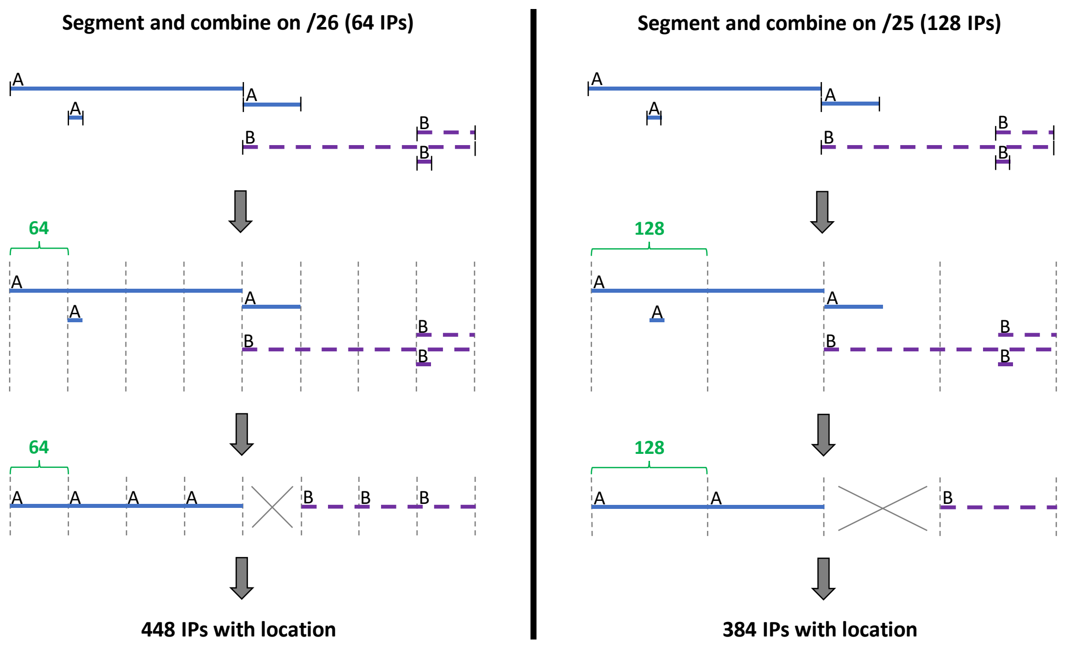

Segment and combine on /26 (64 IPs)
Segment and combine on /25 (128 IPs)
A
B
64
128
448 IPs with location
384 IPs with location

Figure 13.7: Example of the effect that block size has on the result when segmenting and combining records. In both the left and right side we start from the same raw records, each located in either location A which is represented with a blue horizontal solid line, or location B which is represented by a violet horizontal dashed line. In the left example we segment and combine records using a block size of 64 IPs, while in the right example we use a size of 128 IPs. If there are location disagreements, we drop that block from the output. Perhaps counter-intuitively, the example shows that a larger block size can yield lower IP coverage.

13.4 Evaluation

To evaluate the geolocation data extracted from WHOIS datasets, we used a ground truth set of 70 million IP addresses with known location, compiled during the 28-day period ending on October 26th, 2018. Since in this dissertation we focus on worldwide IP geolocation, we evaluated a dataset which combines the data from five WHOIS databases: ARIN, RWhois, AFRINIC, APNIC, LACNIC. As previously discussed, we did not include data from the RIPE NCC database, which does not contain location information.

We experimented with three different parameters. First, we tested different maximum values for **source** IP range sizes. In Figure 13.6 we have observed that as the network range size increases in raw source records, the overall location accuracy decreases. This was expected, as larger ranges cover bigger geographic regions. Therefore, we define the source range size as the maximum size of raw network blocks used as input. If a record has a larger block size, we skip it. Second, we performed experiments using different **target** block sizes used for segmenting IP ranges. Third and finally, we also experimented with the minimum number of location data points required inside a target block in order to obtain a location **consensus**.

To better understand these three parameters, let us discuss an example where we choose a maximum source IP range of 1,024, a target IP range size of 256 addresses, and a minimum of two data points required for location consensus. These parameters mean that among all of the raw WHOIS records, we only use those that have a raw IP range of 1,024 IPs or smaller. We then segment all of these raw IP ranges to fit in smaller IP ranges with a target size of 256 addresses. Finally, we retain only the target IP ranges of 256 IPs which match at least two WHOIS records, and the locations in those records are at most 25 kilometers apart.

Figure 13.8 displays cumulative error distance for a varying number of maximum sizes for the source IP range, a target segmentation block size of 128 IPs, and a minimum of one data points per block used for location consensus. Note that using a minimum of one data point for consensus still implies that if the target block matches more than one data point, all of the matching data points need to be within 25 kilometers of each other. Figure 13.9 similarly shows cumulative error distance, but for a target size of 256 IPs. From both figures we can clearly see that as the maximum source size increases, the overall accuracy goes down. For example, in Figure 13.8, using a source size of 65,536 IPs yields an accuracy of only 13.98% at 10 kilometers, while using a source of 1,048 IPs results in a much improved accuracy of 37.68% at 10 kilometers.

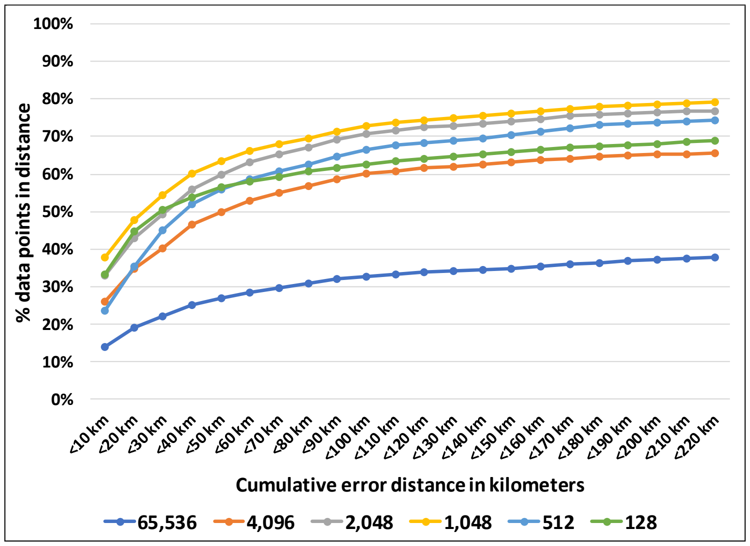


Figure 13.8: Error distance when setting the maximum source IP range size between 65,536 and 256 IPs, setting the target size to 128 IPs, and using a single location as the minimum for location consensus.

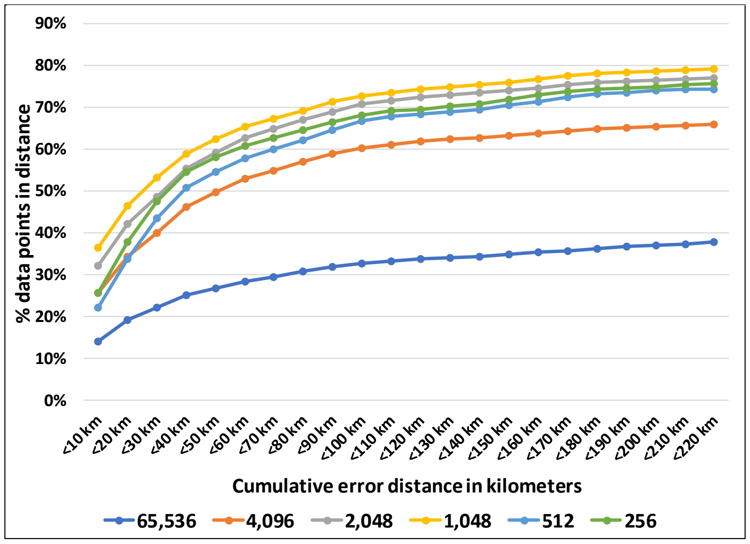


Figure 13.9: Error distance when setting the maximum source IP range size between 65,536 and 256 IPs, setting the target size to 256 IPs, and using a single location as the minimum for location consensus.

However, although reducing the source size results in increased accuracy, it also decreases IP coverage. Using the same example, we obtain a total IP coverage of 881.5 million IP addresses for a source size of 65,536 IPs, and only 109.5 million IP addresses for a source size of 1,024 IPs. Therefore, changing the maximum source size allows tuning the tradeoff between accuracy and coverage.

We will now add a second dimension and discuss the impact that using different values for both the source and target parameters has on accuracy and coverage. Table 13.2 lists the percentage of ground truth data points where the error is smaller than 10 kilometers. The results in the table lead to several observations. First, while varying the maximum source size has a significant impact on accuracy at 10 km, varying the target range size has a smaller impact on the results. For instance, for a maximum source size of 65,536 IPs, varying the target size from 512 to 32 IPs results in little change in accuracy, with a minimum of 13.64% and a maximum of 14.01% error distance at 10 km, respectively. Second, increasing the maximum source size does not always lead to a decrease in accuracy. If we compare the results of maximum source 512 with maximum source 1,024, we can see that the accuracy actually increases. The likely reason for this apparent anomaly is that in Figure 13.6 we have shown that ARIN has comparatively good accuracy for record sizes of 1,024, when compared to record sizes of 512. Since ARIN has the highest coverage among databases, this result is therefore expected. Third, we observe that setting the maximum source to 1,024 always yields the best results, regardless of the value at which we set the target.

Table 13.3 shows the median error for the same combinations of parameters. Here we again observe that we obtain the best results when using a maximum source of 1,024 IPs. Furthermore, using a maximum source of 65,536 IPs leads to high median errors on the order of hundreds of kilometers.

Table 13.4 contains Root Mean Squared Error results, which is a metric that also denotes accuracy. However, as we described in more detail in Chapter 8, RMSE is much more heavily swayed by outliers with high error. The results in the table suggest that WHOIS geolocation data sometimes results in data points with large error in the order of hundreds or thousands of kilometers, which significantly influences RMSE scores. Interestingly, we obtained the best (lowest) RMSE scores when setting the maximum source to 512 IPs. This result shows that using a maximum source of 1,024 IPs might not universally

Table 13.2: Percentage of matching ground truth IPs with an error smaller than 10 kilometers. These numbers are equivalent to the *<10 km* column in Figures 13.8 and 13.9. *Max Source* represents the maximum size of IP ranges in raw WHOIS records that we use. If a record contains a larger network block, we ignore it. *Target* is the fixed size IP range into which we split the source records. For example, if a record contains a network block of size 1,024, we split it into two records for target 512. The table shows that using a max source of 1,024 yields the highest (best) results.

Max Source ↓ Target →	**512**	**256**	**128**	**64**	**32**
65,536 (2^{16})	13.64%	13.93%	13.98%	13.96%	14.01%
4,096 (2^{12})	24.26%	25.55%	25.98%	25.87%	25.61%
2,048 (2^{11})	29.77%	32.04%	32.89%	32.72%	32.15%
1,024 (2^{10})	**32.97%**	**36.41%**	**37.68%**	**37.28%**	**36.44%**
512 (2^{9})	18.85%	22.08%	23.75%	23.16%	22.42%

Table 13.3: The median error distance in kilometers for various combinations of maximum source block size and target block size. The table shows that using a max source size of 1,024 results in the lowest (best) median errors.

Max Source ↓ Target →	**512**	**256**	**128**	**64**	**32**
65,536 (2^{16})	762.2	757.4	754.9	753.6	749.6
4,096 (2^{12})	51.4	51.2	50.0	50.0	51.7
2,048 (2^{11})	33.6	32.0	30.9	31.2	31.9
1,024 (2^{10})	**28.8**	**24.8**	**23.0**	**23.4**	**24.6**
512 (2^{9})	42.0	38.3	35.8	35.9	38.0

be the best choice.

In Table 13.5 we list total IP coverage. Depending on the combination of maximum source and target sizes, we can obtain a coverage between 79.4 million IP addresses on the low end, and 894.7 million IP addresses on the high end. While the results in the previous tables used ground truth data for evaluation, this table lists total IP coverage before the intersection with ground truth. When analyzing the results from all tables, we conclude that obtaining a higher coverage comes at the cost of lower accuracy. On the higher end, using WHOIS data for geolocation yields the highest IP coverage across all the geolocation methods discussed in this dissertation.

Finally, we have also experimented with varying the minimum number of data points needed to make a location consensus decision. For all the previous experiments in this section we set this number to one. When we instead set

Table 13.4: RMSE, which stands for Root Mean Squared Error, across multiple combinations of maximum source block size and target block size. Interestingly, here we see that a maximum source block of 512 yields the best RMSE scores.

Max Source ↓ Target →	**512**	**256**	**128**	**64**	**32**
65,536 (2^{16})	2,636.3	2,643.8	2,652.2	2,641.5	2,639.7
4,096 (2^{12})	1,661.9	1,671.8	1,696.8	1,675.1	1,666.6
2,048 (2^{11})	1,699.6	1,715.9	1,750.2	1,721.1	1,711.0
1,024 (2^{10})	1,780.3	1,794.2	1,842.5	1,800.7	1,792.6
512 (2^{9})	**1,454.0**	**1,449.7**	**1,409.7**	**1,412.4**	**1,381.1**

Table 13.5: Total IP coverage for different combinations of maximum source block size and target size. To obtain IP coverage, we multiply the target IP range size by the number of ranges in the output. Note that unlike the other tables, this table does not depend on ground truth data, as it shows the total coverage.

Max Source ↓ Target →	**512**	**256**	**128**	**64**	**32**
65,536 (2^{16})	**845.3M**	**875.0M**	**881.5M**	**888.0M**	**894.7M**
4,096 (2^{12})	173.0M	178.4M	180.5M	184.0M	188.1M
2,048 (2^{11})	126.4M	129.0M	130.5M	133.6M	137.3M
1,024 (2^{10})	107.2M	108.1M	109.5M	112.4M	115.6M
512 (2^{9})	79.4M	78.9M	80.0M	82.7M	85.5M

the minimum to two data points, we observed that while the accuracy at 10 kilometers improved between 1 and 21 percentage points depending on the maximum source size, the IP coverage was severely impacted. For example, for a maximum source size of 65,536 IPs and a target of 256 IPs, changing the minimum number of location consensus data points from one to two resulted in an increase of 20.7 percentage points for accuracy at 10 kilometers, while at the same time the total IP coverage dropped by 99.2%.

13.5 Conclusions

In this chapter we used location information extracted from WHOIS databases for IP geolocation. To the best of our knowledge, using entire WHOIS databases has never been attempted before in academic work. Also, no related work has systematically evaluated the accuracy and IP coverage of WHOIS data using a ground truth dataset of 70 million IP addresses. We presented an approach to extract, normalize, combine, and segment data from raw network

WHOIS records of different sizes. We experimented with different parameter values for the maximum source sizes and target sizes. The results show that depending on the application, we can tune the results for high coverage or high accuracy. We obtained median error results between 23 kilometers and 762.2 kilometers, depending on the parameters; we also obtained a total IP coverage between 79.4 million IP addresses and 894.7 million IP addresses. Compared to other approached described in this dissertation, using WHOIS data has moderate accuracy results, and much higher total IP coverage. **In Chapter 14, we will further compare the accuracy of this approach with other geolocation methods and with commercial IP geolocation databases.** We will also combine all approaches from this dissertation into a single larger database.

Chapter 14

Conflating IP Geolocation Information

As a final step in our journey of assembling an IP geolocation database from scratch, in this chapter we aim to conflate data from multiple geolocation approaches. Our challenge is to combine the output of several geolocation methods by resolving any location conflicts. For instance, if for a given IP range we have contradictory location information from three approaches, we have to determine which of the three location candidates is likely to be the correct one.

To compile a combined geolocation database, we first train and test each individual geolocation approach using a common ground truth set of 70 million IP addresses. Second, we determine the overlap and location agreement between all pairs of geolocation methods. Third, we combine the output of these approaches by casting the task as a machine learning problem, where for a given IP range, a multiclass classifier decides the best location candidate by choosing among geolocation approaches. Fourth, we determine the contribution that each data source and each feature class has on the accuracy of the classifier. Fifth, we perform an extensive evaluation on the combined database by presenting multiple quality metrics and by comparing it against two state of the art commercial geolocation services. We show that our combined dataset significantly outperforms commercial geolocation services in accuracy. Also, its IP coverage is much higher than previous academic work in IP geolocation.

Throughout this section we will use the metrics described in Table 14.1 to quantify accuracy and coverage.

Table 14.1: Description of metrics used to measure conflation accuracy and coverage, in the order in which they appear in this chapter.

Metric	**Explanation**
Median error	We define error distance for an IP address as the distance between the location estimated by a geolocation approach, and its actual location as given by the ground truth dataset. Median error is the 50th percentile of these measurements, which means 50% of the error distances are smaller than this value.
% error <10km	Percentage of test set data points where this approach made a decision, and the error is smaller than 10 kilometers. We first intersect the ground truth test set with the output of a geolocation method. Then we compute the error distance for each of the IP addresses in the intersection. This metric then is the percentage of these IP addresses where the error is smaller than 10 kilometers.
RMSE in km	Root Mean Squared Error or RMSE is an evaluation metric which measures the differences between values predicted by a model and the actual correct values as defined by a ground truth set [42]. To obtain its value, we take the square root of the mean of the squares of the deviations. One difference between RMSE and median is that RMSE easily gets swayed by large outliers, whereas median does not.
% GT err <10km	Percentage of data points from the **entire** ground truth set - as opposed to the intersection - where the location estimated by this approach has an error smaller than 10 kilometers. Whereas the *% error <10km* metric focuses on accuracy, *% GT err <10km* combines both accuracy and coverage.
Total IP Coverage	The total number of IP addresses for which a particular geolocation approach can output location estimates. This number is independent of the ground truth set.
Weighted TP Rate	The *Weighted TP Rate* metric represents the weighted True Positive rate. This metric is computed by taking the weighted average of the True Positive Rate of each of the classifier classes. The True Positive Rate measures the proportion of actual positives that are correctly identified as such. The weight is given by the number of data points in each class.
Correctly Classified	The percentage of testing data points that were correctly classified. This value is obtained by dividing the number of data points where the overall classifier makes a correct decision, over the total number of data points where the classifier makes any decision, across all classes.
GT Coverage	Ground Truth Coverage Percentage. The percentage of the ground truth dataset for which this particular approach was able to output a location.

14.1 Overlap and Agreement of Approaches

To combine and evaluate the seven geolocation approaches presented in this dissertation, we used a ground truth set of 70 million IP addresses with known location, compiled from Bing logs during the 28-day period ending on October 26th, 2018.

Table 14.2 lists the geolocation methods, along with the Section that describes each approach, as well as accuracy metrics and total IP coverage. In previous chapters we evaluated four of our seven geolocation methods using this more recent ground truth set (*GeoClicks-GPS*, *GeoClicks-Index*, *QueryLogs-V2*, and *WHOIS*). However, we originally evaluated the three remaining methods using older truth sets from 2017 and 2018. To bring all approaches on the same level playing field, we trained and re-evaluated the remaining three geolocation approaches (*ReverseDNS*, *IPInterpolation*, *Traceroute*) with the updated ground truth set, also using ten-fold cross-validation. We used a netmask of /24, which was the IP range size used in most approaches. Wherever applicable, for all geolocation methods we picked the conservative variant that aimed for higher accuracy at the expense of coverage. Nevertheless, the last line in the table shows that the union of all approaches achieves a total IP coverage of close to 600 million addresses, which is higher than most previous work in this area.

The table shows that the accuracy of the approaches varies considerably. The IPInterpolation approach has the best median error at only 2.4 kilometers, while the WHOIS method has the worst median error at 24.8 kilometers. In terms of IP coverage, the Traceroute data source has the lowest coverage at 1.4 million IPs, while IPInterpolation has the best coverage at 357 million IPs.

The values in the *% error <10km* column represent the percentage of IP addresses where error distance is smaller than 10 kilometers. This percentage varies from 23.1% for ReverseDNS, up to 96.2% for IPInterpolation. Whereas the *% error <10km* metric focuses on accuracy, the values in the *% GT err <10km* column combine both accuracy and coverage. The results for this metric are even more disparate, with the Traceroute approach yielding less than 1%, and the QueryLogs-V2 method achieving 45.2%. In this combined metric

Table 14.2: Summary of the geolocation approaches presented in this dissertation. We obtained the accuracy metrics for all methods using ten-fold cross-validation over the same ground truth set of 70 million IP addresses.

	Chapter Section	Median error km	% error <10km	RMSE in km	% GT err <10km	Total IP Coverage
ReverseDNS	8.4	21.9	23.1%	842.7	12.9%	37,156,096
GeoClicks-GPS	9.5.1	4.5	72.2%	893.4	2.2%	19,582,720
GeoClicks-Index	9.5.2	9.2	54.0%	1327.4	10.9%	118,515,200
QueryLogs-V2	10.1	7.6	63.3%	467.7	45.2%	210,220,544
IPInterpolation	11.2	2.4	96.2%	306.1	44.7%	357,289,728
Traceroute	12.3	8.9	55.6%	706.8	1.0%	1,409,792
WHOIS	13.3	24.8	36.4%	1794.2	1.2%	108,136,448
IP Ranges Union						**597,726,720**

the query-based approach interestingly surpasses the one based on IP interpolation, although the former has a much worse median error. The difference in accuracy between the methods is also apparent in Figure 14.1, which shows their cumulative error distance.

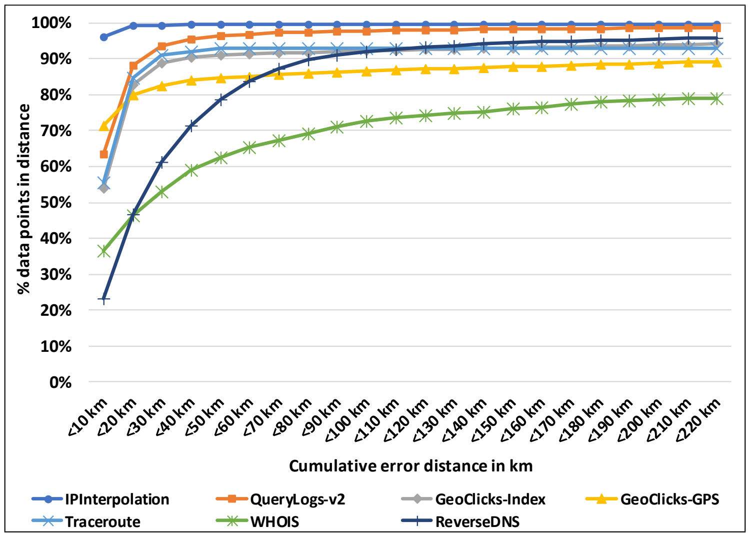


Figure 14.1: Cumulative error distance for each individual IP geolocation approach.

We next investigated the location agreement between all pairs of geolocation approaches. For a given IP range, if location information was available from two or more data sources, we computed the pairwise distance between the location candidates. If a pair of data points was within 25 kilometers of each other, we considered that there is location agreement between them. We decided to use this threshold since we previously used it as a training threshold in several previous chapters. While two locations that are 25 km of each other can be in two different cities, they are still relatively close by and can help determine coarse location agreement in aggregate.

Examining Table 14.3, which contains the agreement between each pair of data sources, leads us to several interesting findings. First, the agreement between the IPInterpolation and GeoClicks-GPS is 87.2%, while the agreement between IPInterpolation and GeoClicks-Index is higher at 94.4%. These results are surprising because both the IPInterpolation and GeoClicks-GPS approaches derive their training data from the GPS ground truth set, while the GeoClicks-Index dataset uses information from the body of clicked documents. One would expect that there would be higher agreement between these two GPS-based methods than between the interpolation approach and index-based method. At the same time, the GeoClicks-Index approach has lower accuracy than GeoClicks-GPS, while IPInterpolation has the best accuracy of all seven approaches. This confirms our finding from Table 9.4 in Chapter 9 that the click-based approaches are complementary, since their overlap is only about 50%. Second, the location agreement between IPInterpolation and QueryLogs-V2 is 98.6%. Having these two methods agree to this extent is unexpected because the query-based approach uses information mined exclusively from user queries which include explicit locations. Therefore, the query-based approach does not use GPS location data, even indirectly. Third, the Traceroute approach has no overlap with the IPInterpolation method, most likely because it has such low overall coverage at only 1.4 million IP addresses.

14.2 Baseline Conflation

In order to determine if using a more complex classifier yields any improvements over a simpler approach, we first created a manual conflation baseline. To assemble the baseline, we followed a simple priority order, by sorting the approaches in ascending order by the median error shown in Table 14.2. For example, if an IP range contained location candidates from both the IPInterpolation and GeoClicks-GPS approaches, we would pick the IPInterpolation candidate since it has a lower overall median error. The implication of using this simple heuristic is that we always make a decision, if an IP range has at

Table 14.3: Location agreement between pairs of geolocation approaches. For an IP range where locations from both items in a pair are available, we compute the distance between the candidates. For the purposes of this table, a pair of data sources are in agreement if their location candidates for an IP range are within 25 km of each other. The table displays the percentage of shared candidates where there is agreement. We specifically discuss the agreement results shown in bold in the text of this section.

	Reverse DNS	**GeoClicks GPS**	**GeoClicks Index**	**QueryLogs V2**	**IP Interp.**	**Trace route**	**WHOIS**
ReverseDNS	—	41.8%	81.2%	65.8%	62.9%	66.7%	59.8%
GeoClicks-GPS	41.8%	—	76.7%	73.9%	**87.2%**	70.7%	67.6%
GeoClicks-Index	81.2%	76.7%	—	95.8%	**94.4%**	73.3%	55.6%
QueryLogs-V2	65.8%	73.9%	95.8%	—	**98.6%**	70.0%	57.1%
IPInterpolation	62.9%	**87.2%**	**94.4%**	**98.6%**	—	**0%**	60.7%
Traceroute	66.7%	70.7%	73.3%	70.0%	**0%**	—	54.1%
WHOIS	59.8%	67.6%	55.6%	57.1%	60.7%	54.1%	—

least one location candidate.

The results yield a median error of 3.7 kilometers and a Root Mean Squared Error of 604.8. Furthermore, 78.9% of data points that intersect with the ground truth dataset are within 10 kilometers of their actual location. Even using such a simple conflation scheme, these results surpass previous work in both accuracy and IP coverage, which is 597.7 million addresses.

14.3 Approach

Now that we have established a strong manual baseline for conflation, we will attempt to create a more accurate conflation model. We propose casting the task of conflating IP geolocation information as a machine learning problem. We train a multi-class classifier to choose among potentially conflicting location candidates, for each range.

14.3.1 Features

The manual baseline makes a decision for every IP range where we have a location candidate from at least a single data source. The union of all approaches covers 597.7 million IP addresses. **The IP coverage of the baseline therefore forms an upper bound of coverage for our improved model.** The implication is that while our more complex model might achieve higher accuracy, it can never surpass the manual baseline in IP coverage. To build a better model, we propose the following categories of features:

- **Candidate Exists**. This category of features simply indicate if a location candidate is available (present) for each data source type. For example, if for a given IP range we have location information both from the ReverseDNS and WHOIS data sources, then the *ReverseDNSPresent* and *WHOISPresent* features would be set to *true*, while the equivalent features for the other five data sources would be set to *false*. The intuition behind this category of features is that during training a classifier can learn the prior probability of each data source being correct.
- **Location Consensus**. The location consensus features count the degree to which the approaches agree on the location of each network block. Each of the seven approaches have a corresponding consensus feature which counts how many other data sources agree with it. By agreement we mean that the location candidate from one data source is within 25 kilometers of the location candidate of another data source. For example, let us say that for a network block we have information from ReverseDNS, IPInterpolation, and Traceroute. If after computing the pairwise distance between them we determine that each pair is within 25 kilometers of each other, then values for the *ReverseDNSConsensusCount*, *IPInterpolationConsensusCount*, and *TracerouteConsensusCount* would all be 2, since each data source agrees with two other data sources. The value of this feature type indicates if multiple data sources agree with each other on what the location for the network block should be.
- **Cluster Confidence**. The clicks and query log methods rely on a variant of the DBSCAN clustering algorithm. In each of these approaches for an IP range we use the location of the largest cluster, if the confidence is above a certain threshold. We define confidence as the number of underlying coordinates that make up the cluster, over the total number of coordinates in that IP range. So if for an IP range the largest cluster contains 23 data points, and the entire IP range contains 30 data points, then the confidence of this largest cluster is $23/30 = 0.77$. The confidence value represents how prevalent the cluster centroid location is among all the coordinates in the

range. A higher confidence might indicate the chosen location is more likely to be correct.

- **Data Points Count**. This feature class records the number of data points that were used by an approach to determine the top location candidate for an IP range. For instance, for ReverseDNS the value of the feature represents the number of hostnames from which we extracted the top candidate location, and for IP Interpolation the value is the number of IPs in the network block that were used to assign the location to the entire block. The intuition behind this type of feature is that if a larger number of separate data points agreed on a location, then the probability that the location is correct increases.
- **Outlier Count**. This feature class is similar to the Cluster Confidence one in that it relies on output from the modified DBSCAN clustering algorithm. For a given set of points, the output of the algorithm contains a list of clusters, and potentially a set of outlier points which are not part of any of the clusters. We decided to try using the number of these outlier points as a feature. A higher number of outlier points could indicate that the data points used to make a judgment for an IP range are spread out over a large geographical region.
- **Other Features**. All of the features in this category are obtained from the output of the WHOIS approach:
    - **Location Extraction Confidence**. When extracting location data from WHOIS databases we determine locations for an IP range by computing the pairwise distances between all the WHOIS records that match that network block. If all of the pairs are within a certain threshold distance of each other, we output the centroid as the likely location for that IP range. We obtain these underlying data points that are used to compute the centroid by using a variant of the Bing reverse geocoding API [69]. For a given string, this API attempts to extract a location. For each such extracted locations we receive a confidence score. We set the *Location Extraction Confidence* feature to the maximum confidence value of any of the underlying individual data points. We use this value as a proxy of the confidence of the entire centroid.

- **Original Source Range Size**. As previously discussed, WHOIS records contain network blocks that can overlap partially or completely. We split these ranges by a target range size. This feature records the maximum size of the original range sizes. The intuition behind this feature is that if the data point originates from a large IP range size, its location is likely to be less correct than if it originates from a smaller network block.

- **Location Candidate Global Occurrences**. Some Internet Service Providers cover large geographical areas, sometimes even spanning an entire continent. When these ISPs receive IP address allocations, they sometimes report the address of their corporate headquarters, regardless of the location of where they actually use these addresses. To mitigate this problem, we count the global occurrences of each raw address in the WHOIS database. If the same identical address appears in the records of multiple IP ranges, the classifier could assign a lower confidence to that location candidate. This feature therefore aims to capture this intuition by storing for each IP range the maximum number of times we have seen the address of any underlying data point. A side benefit of this feature is that it naturally also determines commonly reused fake addresses such as *Fake Street* that we have not already covered by the manual black list previously shown in Figure 13.3.

- **Source Database Name**. We have shown in Figure 13.6 that WHOIS databases have difference accuracy characteristics. This feature contains the name of the WHOIS database from where this location candidate originates. Our assumption is that the classifier can use this name to adjust to the characteristics of each source WHOIS database.

Table 14.4 presents a summary of the features. A checkmark at the intersection of an approach name and a feature type name indicates that we extract that type of feature from that approach. Each checkmark represents a single feature, except for the last checkmark in the table which covers all four features in the *Other* category. This results in 30 distinct features, which are all used together in a single classifier.

Table 14.4: Categories of features available for each IP geolocation approach. Each check-mark except the last one represents one specific feature. Including the four features represented by the last checkmark, there are 30 features in total. All of these features are used **together** in the final conflation classifier.

	Candidate Exists	**Location Consensus**	**Cluster Confidence**	**Data Points #**	**Outlier Count**	**Other Features**
ReverseDNS	✓	✓		✓		
GeoClicks-GPS	✓	✓	✓	✓	✓	
GeoClicks-Index	✓	✓	✓	✓	✓	
QueryLogs-V2	✓	✓	✓	✓	✓	
IPInterpolation	✓	✓		✓		
Traceroute	✓	✓		✓		
WHOIS	✓	✓				✓

14.3.2 Training

We train the classifier on the union of the location candidates from all geolocation methods. For each IP range that matches at least one approach, if an approach has any location candidate, we output the feature values for that approach, or we output null values for the features if that approach does not propose any candidate. For example, if for a particular network block four out of seven approaches output location candidates, we populate the features corresponding to those four approaches, while we set the approaches for the remaining three approaches to null. **The input of the classifier is the feature vector for an IP range and the output is the class name that indicates which geolocation approach has the most likely location candidate.** The classifier has eight classes, one for each of the names of the seven approaches, and a last class called *None*, which signifies that we should not pick any of the location candidates for that block.

In order to train this classifier we attach class labels to the dataset we obtained by taking the union of network blocks covered by the seven geolocation approaches. We then intersect this dataset with the ground truth set. For each IP range in the set we retain the location candidates generated by the geolocation approaches, as well as the actual correct location from the ground truth set. We then compute the distance between each geolocation approach and the correct location. We set the class label to be the name of the geolocation approach that is closest to the ground truth location. If none of the distances comes within 25 kilometers of the actual location, we set the class label to *None*.

14.3.3 Classifier Type

To determine the optimal classifier, we ran experiments using multiple types of widely used classifiers. **Naïve Bayes** classifiers use Bayes' theorem, which describes the probability of an event, based on prior knowledge of the features that might be related to the event [180]. **Logistic Regression** classifiers are statistical models that use a logistic function to model a binary variable. The logistic functions has a common *"S"* shape sigmoid curve. The training process uses data to estimate the parameters of the function [181]. The **C4.5 algorithm** uses the concept of information entropy to train a decision tree. At runtime the tree can be quickly followed to reach a decision [182]. A **decision table** consists of a hierarchical table in wich each entry in a higher level table gets broken down by the values of a pair of additional attributes to form another table. These tables can also be expressed as decision trees [183]. **Support Vector Machines** is another well-known type of binary classifier that learns a hyperplane boundary between training data points [184]. During training it attempts to minimize the geometric distance between the two classes.

We also experimented with two meta-algoritms: **Ada Boost** [185] and **Rotation Forest** [186]. These algorithms combine the individual decisions of multiple classifiers to create an ensemble of classifiers that together make a final decision. In Ada Boost the classifiers are added one at a time so that each subsequent classifier is trained on data which was deemed "hard" for the previous ensemble members. In each such step misclassified input data gains a higher weight and examples that are classified correctly lose weight. Rotation Forest is different in that it generates multiple smaller classifiers by splitting the feature set randomly into sets. In each set a subset of features is selected using Principal Component Analysis. Then, multiple smaller classifiers are constructed by rotating which subsets of features are used. For a given input, each smaller classifier outputs a confidence for each class. The confidence per class is then averaged across all classifiers and the class with the highest

Table 14.5: Comparison of the accuracy obtained from different types of classifiers.

	Weighted TP Rate	**Correctly Classified %**	**% err <10km**	**Median Error**	**RMSE in km**
Naïve Bayes [180]	79.50%	79.47%	78.52%	3.82	510.08
Logistic Regression [181]	84.30%	84.27%	80.62%	3.60	492.24
C4.5 Decision Trees [182]	84.30%	84.30%	80.72%	**3.59**	473.70
Decision Table [183]	84.20%	84.19%	80.73%	**3.59**	477.76
Support Vector Machines [184]	84.20%	84.21%%	80.52%	3.61	492.61
Ada Boost w/ C4.5 [182, 185]	82.10%	82.08%	80.63%	3.63	**470.66**
Rotation Forest w/ C4.5 [182, 186]	**84.40%**	**84.36%**	**80.78%**	**3.59**	483.53

confidence is chosen.

Table 14.5 lists the results across the different classifiers. Most classifier types achieve a reasonable accuracy, with three of them reaching a median error of only 3.59 kilometers. With the exception of the Naïve Bayes classifier which has the worst performance, the classifiers yield very similar results. To analyze why that is, we examined the decision tree generated by the C4.5 classifier. One property of decision trees is that they emit human readable rules. The root of the tree makes a decision on the existence of the IPInterpolation location candidate, which is the geolocation approach with the best accuracy and highest total IP coverage. The second level of the tree makes decisions on the existence of the QueryLogs-V2 and GeoClicks-GPS approaches. These three geolocation methods have the best median error, and they are also the first three approaches we use in our manual rules baseline. The difference however is that the left and right branches on the second level make decisions on different approaches, while our manual rules list uses a priority order. As we get closer to the leaf nodes, more categories of features are used. The decision tree contains 573 nodes and 290 leaves. In conclusion, the reason most classifiers do a reasonable job at picking the right class label is that there are several approaches with high accuracy, and their mere presence is a good enough signal.

Furthermore, we also examined the confusion matrix of the decision tree. We observed that the pair of methods with most confusion are the IP interpolation and query-based ones. In Table 14.3 we have shown that the agreement between these two methods is very high, at 98.6%. Therefore, it is expected that the classifier is more confused as to which of the two labels to pick from, since we only output a single class. The downside of this multi-class design is that we always output a single class, even if two or more geolocation approaches emit candidates that are close to the actual location from ground truth. During training we always just pick the approach which is closest to the ground truth IP location. This has a negative impact on true positive rate, but does not have a significant negative impact on median error or RMSE, because picking either of the two labels would result in a small error.

The results show that Rotation Forest meta-algorithm with C4.5 decision trees is the best overall classifier. Therefore, in the remainder of this chapter we will perform all experiments using this type of classifier.

14.4 Preliminary Evaluation

As a preliminary evaluation of the classifier we will compare the manual baseline with the machine learning approach that we name *SearchGeo*. Table 14.6 presents both accuracy and coverage metrics. Compared to the accuracy of the manual baseline, the classifier is better in the percentage of ground truth IPs where error is smaller than 10 kilometers, in median error, and in Root Mean Squared Error. This indicates that the additional features used in the classifier can indeed achieve higher accuracy, which is what we set out to achieve.

We previously mentioned that the manual baseline represents an upper bound in terms of IP coverage. The reason is that in the manual baseline we make a decision for **every** network block covered by any of the geolocation approaches. In contrast, the SearchGeo classifier can also output a *None* class, which indicates we should not make a decision for an IP range, which in turn reduces the overall IP coverage. The table shows that, as expected, the IP coverage for the classifier is lower, but only by about 2.8%. The results show that we can achieve higher accuracy, at the expense of more complexity and slightly lower coverage. If IP coverage is more important than accuracy, then using the manual priority approach might be a reasonable approach.

Figure 14.2 also shows the difference in accuracy between the two models. The error distance curves reveal that the SearchGeo machine learning approach has higher accuracy than the manual one across the entire error distance range.

Table 14.6: Comparison of accuracy and coverage metrics between the manual baseline and and our *SearchGeo* machine learning approach. The *GT Coverage* column displays the percentage of the ground truth dataset which is covered by the approach.

	% error <10km	**Median error**	**RMSE in km**	**GT Coverage**	**Total IP Coverage**
Manual Baseline	78.93%	3.71 km	604.76	**70.7%**	**597,726,720**
SearchGeo	**80.78%**	**3.59 km**	**483.53**	68.7%	581,010,688

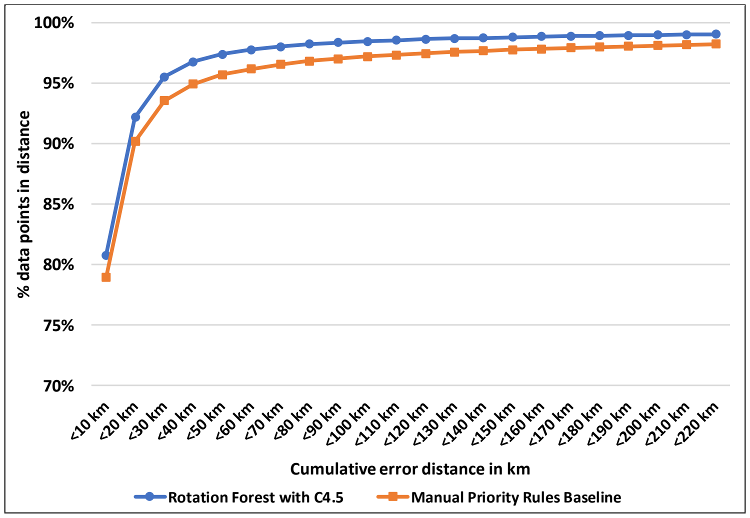


Figure 14.2: Comparison of cumulative error distance between the manual baseline and the decision trees based classifier.

14.5 Feature Analysis

The classifier uses 30 features, segmented across seven geolocation approaches and six feature categories. Some of these features may be more important than others. Here we present an analysis on how important geolocation approaches and feature classes are to the final classifier.

14.5.1 Leave One Approach Out

The results in Table 14.2 have shown that the IP interpolation approach has both the highest accuracy and the best coverage. Our manual analysis of the C4.5 decision tree has revealed that the most important decision at the root of the tree checks if IP interpolation data is available. Therefore, a natural

Table 14.7: Comparison of accuracy metrics when leave one IP geolocation approach out.

	Correctly Classified	Median error	% error <10km	RMSE in km	% GT err <10km	Total IP Coverage
All Geolocation Approaches	84.36%	3.59 km	80.78%	483.53	55.49%	**581,010,688**
All except ReverseDNS	84.88%	3.60 km	80.78%	526.48	55.68%	574,211,840
All except GeoClicks-GPS	85.12%	3.56 km	80.80%	461.16	55.48%	573,776,640
All except GeoClicks-Index	85.96%	3.52 km	81.19%	**376.55**	55.13%	553,672,960
All except QueryLogs-V2	**93.25%**	**2.68 km**	**91.03%**	481.09	46.84%	510,349,568
All except IPInterpolation	84.32%	7.49 km	63.24%	642.75	31.75%	381,070,336
All except Traceroute	84.28%	3.60 km	80.48%	496.11	55.56%	579,872,256
All except WHOIS	84.98%	3.60 km	80.67%	477.39	55.50%	496,244,480

question arises as to what effect would removing the predicting power of an individual approach have on the accuracy of the classifier.

To answer this question, we separately trained seven classifiers by leaving out all features derived from one particular approach. The results in Table 14.7 confirm our prior findings that the interpolation-based geolocation method has the biggest effect on overall accuracy. When we train a classifier with all features except the ones based on the interpolation-based approach, we can see that the median error more than doubles to 7.49 kilometers. However, this result in itself is still better than most prior academic work. This accuracy is also higher than that of commercial services, as we will show later.

An interesting, and perhaps counter-intuitive, finding is that if we train a classifier without the features from the query-based method then the accuracy of the classifier goes up. In fact, removing these features yields the best median error and percentage of ground truth IPs where error is less than 10 kilometers, at 2.68 kilometers and 91.03%, respectively. The reason for this change is that QueryLogs-V2 is the approach with second-highest individual IP coverage and on its own has a higher median error at 7.6 kilometers. Therefore, removing its features allows the IP Interpolation approach, which has better median error, to now cover a larger percentage of the IP coverage. However, the downside of removing these query-based features is that the total IP coverage goes down by tens of millions of IP addresses.

14.5.2 Leave One Feature Class Out

We continue our feature analysis by studying the contribution of feature classes to the overall classifier. We have previously split the features in six feature

Table 14.8: Comparison of accuracy metrics when we leave one feature class out.

	Correctly Classified	Median error	% error <10km	RMSE in km	% GT err <10km
All Features	84.36%	**3.59 km**	80.78%	483.53	55.49%
All except Location Consensus	84.28%	**3.59 km**	80.73%	479.55	55.47%
All except Cluster Confidence	**84.38%**	**3.59 km**	80.73%	484.27	**55.54%**
All except Data Points Count	84.35%	**3.59 km**	**80.80%**	487.42	55.46%
All except Outlier Count	84.34%	**3.59 km**	80.74%	482.19	55.51%
All except Other Features	84.31%	**3.59 km**	80.77%	**476.84**	55.47%

classes. The first class, called *Location Candidate Exists* is a fundamental requirement in our classifier, so we cannot perform experiments by removing it. If we were to remove these features, then the classifier could still derive their values indirectly by the existence of the other features. When a geolocation approach does not have any location candidate, then all of its features are set to null. Therefore the *Candidate Exists* feature is somewhat redundant and we only perform experiments with the other five feature classes.

Table 14.8 shows the effect that removing each feature type has on accuracy. Note that since here we remove feature classes which span all geolocation approaches, the total IP coverage does not change. This is because when we remove a feature class, we do so from all approaches, but the rest of the feature types of the approaches remain. The table shows little change in accuracy metrics when removing any one feature type. It is possible that the feature types are complementary and no particular feature type dominates the other ones. Another likely explanation is that the mere presence of a location candidate by a particular IP geolocation approach provides enough predictive power as to make the other feature types irrelevant. We investigate this second hypothesis in the next section.

14.5.3 Add Feature Types One By One

The final step of our analysis is to show the impact that adding feature types one by one cumulatively has on classifier accuracy and error distance. We use the same feature types as in the previous section, but we now progressively add the feature types, instead of leaving one of them out. Table 14.9 demonstrates that as we add more feature types, the median error reduces, and the percentage of correctly classified data points increases.

To answer the question we posed in the previous section as to the impact

Table 14.9: Comparison of accuracy metrics when we use all IP geolocation approaches and we progressively add feature classes one by one.

	Correctly Classified	Median error	% error <10km	RMSE in km	% GT err <10km
Location Candidate Present	83.92%	3.63 km	80.26%	525.61	55.56%
+ Location Consensus	84.01%	3.62 km	80.27%	525.13	55.58%
+ Cluster Confidence	84.08%	3.61 km	80.44%	519.35	55.49%
+ Data Points Count	84.30%	3.59 km	80.74%	477.63	55.48%
+ Outlier Count	84.31%	3.59 km	80.77%	476.84	55.47%
+ Other Features (All)	84.36%	3.59 km	80.78%	483.53	55.49%

Table 14.10: Comparison of accuracy metrics when we use all IP geolocation approaches **except IPInterpolation** and we progressively add feature classes one by one.

	Correctly Classified	Median error	% error <10km	RMSE in km	% GT err <10km
Location Candidate Present	81.79%	7.62 km	62.39%	760.52	31.30%
+ Location Consensus	83.32%	7.60 km	62.35%	755.79	31.72%
+ Cluster Confidence	83.50%	7.57 km	62.57%	738.55	31.65%
+ Data Points Count	84.05%	7.52 km	63.07%	643.96	31.64%
+ Outlier Count	84.09%	7.50 km	63.16%	643.31	31.65%
+ Other Features (All)	84.32%	7.49 km	63.24%	642.75	31.75%

that the dominant interpolation approach has on accuracy, we perform the same experiment, but we train the classifiers without using any data from the IP Interpolation approach. The results, which are shown in Table 14.10, show that the rate of increase in accuracy when adding feature types is slightly higher. The percentage of correctly classified instances gradually increases by 2.53 percentage points, while in the previous experiment it increased by only 0.44 percentage points. We conclude that the mere presence of the IP Interpolation approach does have some impact across all of the feature types, but the impact is not significant. Therefore, the feature classes themselves do help in achieving more accurate results.

14.6 Commercial Evaluation

Finally, we evaluate our combined approach against two state of the art commercial geolocation services. The results in table 14.11 demonstrate that our combined SearchGeo approach significantly surpasses the two commercial databases across several metrics. The median error of our database is three times better than that of ProviderA, which is the best out of the two commercial baselines. The results for the percentage of ground truth data points where

Table 14.11: Comparison between our approach *SearchGeo* and two state of the art commercial geolocation services, ProviderA and ProviderB, across several metrics.

	Median error	**% error <10km**	**RMSE in km**	**% GT err <10km**
SearchGeo	**3.6 km**	**80.78%**	**483.5**	**55.49%**
ProviderA	11.1 km	47.24%	545.9	47.24%
ProviderB	16.7 km	36.74%	545.3	36.74%

error is smaller than 10 kilometers, as well as Root Mean Squared Error are also considerably better than that of the commercial databases.

One important caveat to these results is that the total IP coverage of the commercial databases is larger than that of our approach. Whereas SearchGeo has a total coverage of about 600 million addresses, the commercial databases cover up to 3 billion IPs. However, our approach does outperform the commercial services when we combine accuracy and coverage. The last column in the table shows the percentage of IPs in the **entire** ground truth set of 70 million addresses where error is smaller than 10 kilometers. Here our combined method is again significantly better than the commercial baselines, even if they have the advantage of higher total IP coverage. The massive difference in accuracy between the approaches is also apparent in Figure 14.3, which shows cumulative error distance.

As we have detailed in Chapter 5, most previous work targets single specific countries. In contrast, in this dissertation we have designed our approaches to work across the entire globe. Figure 14.4 contains a comparison of accuracy at 10 kilometers across several countries of different sizes, and which are located on several continents. In the figure we compare the accuracy of our approach to that of ProviderA, which is the best out of the two commercial baselines. Here our approach again significantly outperforms the commercial baseline. In fact, we have not found a single country where our accuracy is worse than either of the commercial databases.

In conclusion, our machine learning based approach unequivocally surpasses current state of the art commercial geolocation services in accuracy. Its total IP coverage of close to 600 million IP addresses is also higher than

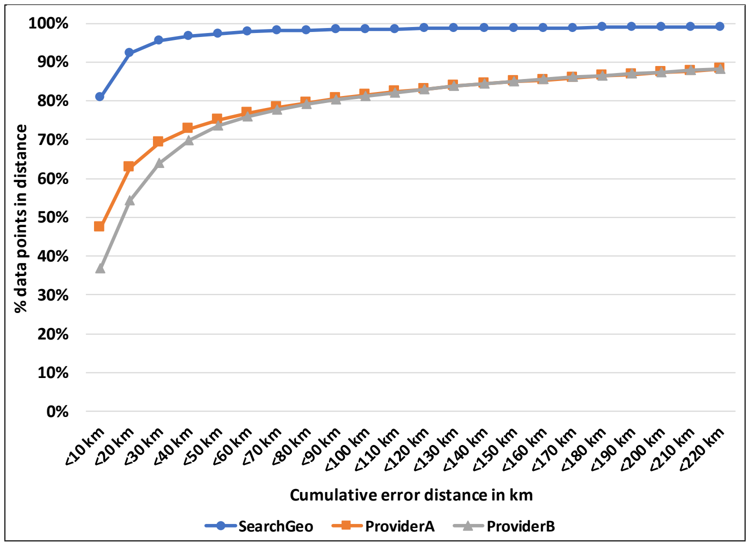


Figure 14.3: Comparison of cumulative error distance between our approach *SearchGeo* and two state of the art commercial geolocation services, *ProviderA* and *ProviderB*.

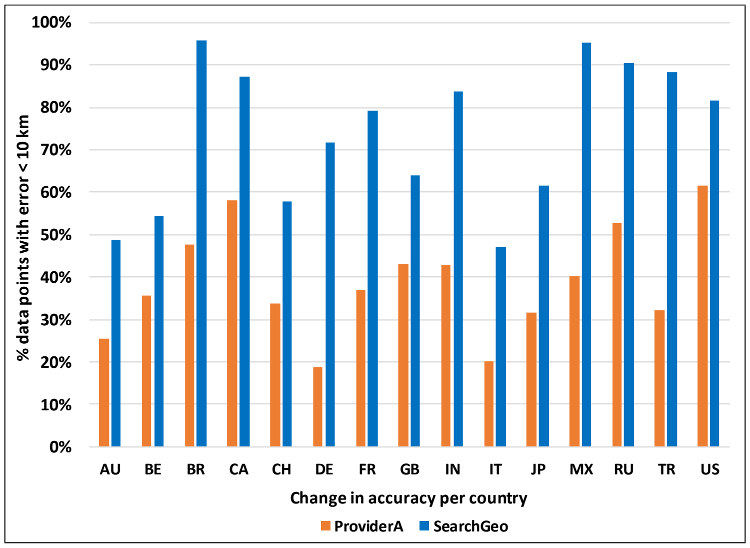


Figure 14.4: Increase in the percentage of data points with error smaller than 10 kilometers between the best commercial approach *ProviderA*, and our approach, *SearchGeo*.

that of previous work in the area of IP geolocation. While the total IP coverage of SearchGeo is not as high as that of commercial services, it still outperforms them in a metric which combines both accuracy and coverage.

Chapter 15

Conclusions and Future Work

The goal of this dissertation was to compile a geolocation database from scratch. To that end, we have developed and evaluated seven methods for IP geolocation. We have placed particular emphasis on approaches which exploit data extracted from search engine logs, as this source of information has not before been used in IP geolocation literature. We have combined these seven approaches to obtain *SearchGeo*, which has a median error of only 3.59 kilometers and a total IP coverage of 581 million IP addresses. Table 15.1 shows a summary of our approaches, compared to two commercial baselines. The results show that SearchGeo achieves much higher accuracy, but it has lower total IP coverage than the commercial baselines. However, using a metric which combines accuracy and ground truth coverage, our approach again outperforms the baselines.

In this final chapter we revisit and answer the research questions posed in the Introduction section. We also propose multiple avenues for future research.

15.1 Revisiting Research Questions

At the onset of our work we set out to answer several research questions pertaining to IP geolocation. Here we again address those same questions, and summarize the approaches we have taken to answer them.

How can we obtain a large ground-truth set for IP geolocation? Throughout this dissertation we have used ground truth datasets obtained by

Table 15.1: Summary of the geolocation approaches presented in this dissertation, compared to two commercial baselines.

	Median error km	% error <10km	RMSE in km	% GT err <10km	Total IP Coverage
ReverseDNS	21.9	23.1%	842.7	12.9%	37,156,096
GeoClicks-GPS	4.5	72.2%	893.4	2.2%	19,582,720
GeoClicks-Index	9.2	54.0%	1327.4	10.9%	118,515,200
QueryLogs-V2	7.6	63.3%	467.7	45.2%	210,220,544
IPInterpolation	2.4	96.2%	306.1	44.7%	357,289,728
Traceroute	8.9	55.6%	706.8	1.0%	1,409,792
WHOIS	24.8	36.4%	1794.2	1.2%	108,136,448
SearchGeo (all conflated)	3.6	80.8%	483.5	55.4%	581,010,688
Commercial Baseline A	11.1	47.2%	545.9	47.2%	3+ Billion
Commercial Baseline B	16.7	36.7%	545.3	36.7%	3+ Billion

mining search engine logs for impressions where the user has opted-in to provide their precise GPS location. Sections 3.1 and 8.5.1 detail how we compiled these data sets. As discussed in Section 6.1, we have taken multiple steps to safeguard the privacy of users, including randomizing each raw location by 200 to 584 meters in a random direction. Between 2014 and 2018 we compiled and used for training and evaluation multiple ground truth datasets, ranging from 8.4 million to 70 million IP addresses in size. To the best of our knowledge, these datasets are the largest ones used in IP geolocation literature. They span all countries in the world and they contain IPs from both mobile and fixed broadband IP addresses.

What is the impact of inaccurate IP geolocation on user experience?

In Chapter 3 we have studied the impact that incorrect IP geolocation has on users, in the context of search engines. By mining Bing logs, we have shown that both overall and ad click-through rates decrease when the location is incorrect by 4.3% and 17.9%, respectively. Prior research has shown that this metric is positively correlated with user satisfaction [45]. Furthermore, we have found that ad revenue also decreased by 40.3%. Conversely, this result suggests that correcting user location could have a positive impact on revenue. To the best of our knowledge, our study is the only one in literature which attempts to quantify the effect of inaccurate IP geolocation.

How accurate are commercial geolocation databases?

Using a ground truth set of 8.4 million IP addresses, in Section 4.2 we have shown that the city-level accuracy of three commercial geolocation services does not exceed 70% in any country. The results also show that in some countries such as UK, France, Italy, Spain, and Germany the accuracy dips below 20%, and sometimes even below 10%. These findings suggest that IP geolocation is still an unsolved problem. In contrast, Section 14.6 demonstrates that our final approach results in accuracy higher than 70% in several countries.

Can we devise new IP geolocation approaches by exploiting information from the logs of a search engine?

We developed several novel IP geolocation approaches that use information mined from search engine logs. First, in Chapters 7 and 10 we extracted explicit locations from user queries such as *"indian food denver"*. By clustering these raw locations by IP range, we were able to achieve a median error of 7.6 kilometers, and an coverage of 210 million IP addresses. Second, in Chapter 9 we propagated locations from IP addresses with known location to addresses with unknown location, through user clicks. The intuition behind this approach was that users who click on search results with local geographic focus, such as on the website of a community theater in Seattle, are in aggregate located in that location. The variant of this approach tuned for higher accuracy obtained a median error of only 4.5 kilometers, with a total IP coverage of 19.6 million addresses. Third, in Chapter 9 we have also developed a click-based approach which instead of propagating users' GPS locations, it extracts physical addresses from the body of clicked web documents. This web index-based approach, also tuned for higher accuracy, yields a median error of 9.2 kilometers, and a much better IP coverage at 118.5 million addresses.

Is it possible to improve IP geolocation coverage by interpolating (extrapolating) locations across neighboring IP addresses and neighboring IP ranges?

In Chapter 11 we have proposed a process to extrapolate the location of an entire IP range by using the locations of a few underlying IP addresses with known location. Here we again use information mined from search engine logs. Specifically, we use the ground truth set that contains IPs with known location and we train and evaluate using ten-fold cross validation. This results in a remarkable median error of only 2.4 kilometers, and a large total IP coverage of 357 million addresses.

Can we use public information such as WHOIS databases and traceroutes to augment IP geolocation?

We propose a systematic approach to parse, normalize, and combine WHOIS databases in Chapter 13. We show that network block size is correlated with median error, with larger blocks resulting in higher error, as they cover bigger geographic regions. We also compare the accuracy characteristics of the different databases. Using a target size of 256 IPs, this approach results in a median error of 24.8 kilometers and a coverage of 108 million IPs. In Chapter 12 we also investigate using traceroute information for geolocation. We introduce the concept of latency neighbors, which are pairs of nodes along a traceroute path that are within a certain threshold of milliseconds from each other. We extract latency neighbors from a dataset provided by CAIDA which contains 9 billion traceroutes. We then propagate locations in the pairs, from neighbors with known location to neighbors with unknown location. A variant of this approach which we tuned for higher accuracy achieves a median error of 8.9 kilometers on the more recent 2018 ground truth set, and an IP coverage of 1.4 million addresses.

Which IP geolocation methods provide the most IP coverage and accuracy? What are the advantages and disadvantages of each method?

In Table 15.1 we summarize the results for the high accuracy variants of each of the seven geolocation approaches, along with the metrics of the combined SearchGeo approach, compared to two commercial baselines. The results indicate that the three approaches based on public data fare the worst in terms of accuracy (ReverseDNS, WHOIS) and IP coverage (Traceroute). Although the Traceroute approach has the lowest coverage at only 1.4 million addresses, the methods based on ReverseDNS and WHOIS achieve higher coverage at 37.2 and 108 million addresses, respectively. Therefore, the advantages of these approaches are that they are based on easily accessible public data, and some of them have good coverage. Their main disadvantage is that they are less accurate than the commercial baselines. The other four methods, which are based on information mined from search engine logs, have the best accuracy, with median error between 2.4 and 9.2 kilometers. The method based on IP interpolation has the highest coverage, with 357.3 million addresses. Their advantages therefore include high accuracy and coverage. Their main disadvantage is that they require access to application logs, which are generally proprietary. A challenge shared by all these seven approaches is that they all require a large amount of data storage and computation. The SearchGeo approach, which combines data from all seven geolocation methods, achieves the highest accuracy for all metrics. Although its total IP coverage is lower than that of commercial geolocation services, it still surpasses its commercial counterparts by the metric which combines both accuracy and coverage.

How can IP geolocation methods be conflated into a larger geolocation database?

Chapter 14 presents our proposed approach on combining the output of the several geolocation approaches into a database which provides a single location candidate per IP range. Since for a network block the approaches can output multiple and sometimes conflicting location candidates, we needed to find a way to resolve conflicts. We cast this conflation task as a machine learning problem. We trained a classifier which for a given IP range can decide which location candidate is most likely correct. The combined approach, named SearchGeo in Table 15.1, achieves good accuracy results and moderate IP coverage.

15.2 Future Work

There are multiple avenues for future work. We begin by presenting multiple approach-specific ways in which our work can be extended. Then, we also propose several suggestions which span multiple approaches.

Approach-specific suggestions:

- **Query Logs**. Both versions of our query logs approach depend on locations extracted from explicit user queries. To parse these locations we use a variant of the Bing reverse geocoding API [69]. Although this API does a reasonable job in parsing locations, a manual verification of its output reveals that it sometimes misses or misinterprets locations. We have seen cases where a street name is interpreted as a city, although the correct city name is present later in the string. We have also seen indications that the reverse geocoder is less accurate when parsing foreign language queries. Therefore, one potential avenue for future work would be to compare multiple types of reverse geocoding APIs, or perhaps even develop a new one from scratch. Potential outcomes of using a more accurate reverse geocoder are higher accuracy and higher IP coverage.

- **Reverse DNS**. The classifier which evaluates location candidates from reverse DNS hostnames uses 11 primary features and 4 secondary features. These features are listed in Tables 8.4 and 8.5 on page 96. The features are extracted from multiple freely available databases. Since the classifier can theoretically match any city in the world, it requires a large amount of memory and CPU resources. We propose two potential ways to improve memory usage, reduce computation complexity, and cut down on the number of location candidates. First, it may be possible to reduce the number of strings that are used to find locations in hostname items. For instance, the data behind the *Alternate names* feature sometimes contains tens or arcane names per city. It may be possible to significantly reduce the number of these strings, which will save on computing resources and potentially reduce the number of false positives. Second, we have observed that not all features are useful in practice. One could potentially remove some of these features, without impacting accuracy too much.
- **Geographic Clicks**. Both the GPS and index-based approaches propagate locations through user clicks. We do not distinguish between the type of page or page element that was clicked. One could further develop these approaches by further breaking down the types of clicks. For example, if a click is issued inside a search result page, did the user click on a simple algorithmic result, or inside an answer module such as business listings, weather, or movie showtimes? Also, does the intent of the user query before the click matter? Furthermore, is there a difference between users that click on news articles, versus people that click on Wikipedia pages? In summary, it might be worthwhile to investigate if specific categories of clicks better reveal the location of users.
- **IP Interpolation**. One direction for future work is to perform IP interpolation on progressively larger network blocks, every time assigning a location to a larger parent block only if at least 50%+1 of the smaller underlying

blocks agree on a location. This would allow increasing the IP coverage and it would also allow outputting blocks of different sizes.

- **Traceroute**. One potential area for improvement would be to exploit more information available in traceroute paths. For instance, it should be possible to parse the reverse DNS hostnames of nodes along traceroute paths to extract more location hints. These hints would then be propagated through latency neighbors, similarly to how we propagated GPS locations.

- **WHOIS**. Using a better reverse geocoder could also improve the accuracy and coverage of the WHOIS geolocation approach. Another area for future work could be to intersect WHOIS locations with BGP network dumps. Previous work has shown that Autonomous Networks which are stubs, that is they have a single backbone connection to a larger ISP, are typically located in the same geographic region [31]. Therefore, if we find the corresponding WHOIS records for these networks, we might be able to assign a higher confidence to their locations.

Overall cross-approach suggestions:

- **Location Granularity**. Locations in geolocation databases typically have city-level granularity. Once IP geolocation becomes precise enough, it might be possible to target other granularities. For instance, since our combined approach has a median error of 3.6 kilometers, it would be interesting to investigate if we can output locations at a neighborhood level.
- **IPv6**. In this dissertation we focus on locating IPv4 addresses. However, all of our proposed approaches could theoretically be applied to IPv6 addresses. Therefore, an avenue for future work would be to investigate if there are any new challenges that arise when we apply our findings to these types of addresses.
- **IP Coverage**. Although our combined SearchGeo approach achieves a total IP coverage of close to 600 million IP addresses, that still falls short of commercial geolocation services, which cover most of the IPv4 space. One potential of improvement would be to investigate how to improve total IP coverage further.
- **Block Size**. Although in this dissertation we have experimented with block sizes to some degree in some chapters, it would be interesting to more systematically study the effect that block size has on geolocation accuracy. For instance, would clustering clicks using a different block size have an impact on accuracy? Furthermore, it might be worthwhile to investigate ways in which the final conflated database can contain blocks of variable size.
- **Network Delay**. In Chapter 5 we have established that network delay based approaches have poor accuracy and coverage, and therefore we avoided using them in this dissertation. However, there is one use case for using network delay information. The RMSE metric is influenced by outliers. If a geolocation approach outputs an incorrect location which is very far away from the true location, then this outlier has an outsized effect on RMSE than, for example, on median error. We could use network delay to reduce the number of outliers. For example, by converting ping latency to distance we could easily confirm if a location extracted from WHOIS databases is plausible. If we issue pings from servers located in the same country as the target IP, converting the resulting latency measurements to distance could indicate whether the country is correct. If instead the latency is much higher

than expected, this could indicate that the location extracted from WHOIS might not be correct.

Bibliography

[1] Huang, C., Maltz, D., Li, J. & Greenberg, A. Public DNS system and Global Traffic Management. In *INFOCOM 2011*, 2615–2623 (2011).

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

[3] Bhatla, T. P., Prabhu, V. & Dua, A. Understanding credit card frauds. *Cards business review* **1** (2003).

[4] Akhilomen, J. Data mining application for cyber credit-card fraud detection system. In *Industrial Conference on Data Mining*, 218–228 (Springer, 2013).

[5] Kotila, M., Rumin, R. C. & Dhar, S. Compendium of ad fraud knowledge for media investors (2016). URL `https://www.wfanet.org/app/uploads/2017/04/WFA_Compendium_Of_Ad_Fraud_Knowledge.pdf`. Accessed: 2019-02-06.

[6] Smith, C. S. Geolocation: Core To The Local Space And Key To Click-Fraud Detection (2007). URL `https://searchengineland.com/geolocation-core-to-the-local-space-and-key-to-click-fraud-detection-11922`. Accessed: 2019-02-06.

[7] Kölmel, B. & Alexakis, S. Location based advertising. In *First International Conference on Mobile Business* (Athens, Greece, 2002).

[8] Huffman, S. M. & Reifer, M. H. Method for geolocating logical network addresses (2005). US Patent 6,947,978.

[9] Maughan, D. *et al.* A roadmap for cybersecurity research. *US Department of Homeland SecurityNovember* **2009** (2009).

[10] Shue, C. A., Paul, N. & Taylor, C. R. From an IP Address to a Street Address: Using Wireless Signals to Locate a Target. In *WOOT 2013* (USENIX, Washington, D.C., 2013). URL `https://www.usenix.org/conference/woot13/workshop-program/presentation/Shue`.

[11] Butkovic, A., Orucevic, F. & Tanovic, A. 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 (2013).

[12] MacVittie, L. Geolocation and Application Delivery. `https://www.f5.com/pdf/white-papers/geolocation-wp.pdf` (2012). Accessed: 2018-08-02.

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

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

[15] Targetly, G. Automatically Switching Website Language Based On Visitor Country (2019). URL `https://geotargetly.com/automatically-switch-website-language-based-on-country`. Accessed: 2019-01-26.

[16] Bennett, P. N., Radlinski, F., White, R. W. & Yilmaz, E. Inferring and Using Location Metadata to Personalize Web Search. In *SIGIR 2011*, 135–144 (ACM, Beijing, China, 2011). URL `http://doi.acm.org/10.1145/2009916.2009938`.

[17] Kliman-Silver, C., Hannak, A., Lazer, D., Wilson, C. & Mislove, A. Location, location, location: The impact of geolocation on web search personalization. In *Proceedings of the 2015 Internet Measurement Conference*, 121–127 (ACM, 2015).

[18] Zhuang, Z., Brunk, C. & Giles, C. L. Modeling and visualizing geo-sensitive queries based on user clicks. In *Proceedings of the first international workshop on Location and the web*, 73–76 (ACM, 2008).

[19] White, R. & Buscher, G. Characterizing local interests and local knowledge. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1607–1610 (ACM, 2012).

[20] White, R. W. *et al.* Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22nd international conference on World Wide Web*, 1411–1420 (ACM, 2013).

[21] Yan, J., Chu, W. & White, R. W. Cohort modeling for enhanced personalized search. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 505–514 (ACM, 2014).

[22] MaxMind, I. Detect Online Fraud and Locate Online Visitors (2018). URL `https://www.maxmind.com/en/home`. Accessed: 2018-08-13.

[23] Neustar, Inc. IP Intelligence (2018). URL `https://www.security.neustar/digital-performance/ip-intelligence`. Accessed: 2018-08-13.

[24] IP2Location.com. Geolocate IP Address Location using IP2Location (2018). URL `https://www.ip2location.com/`. Accessed: 2018-08-13.

[25] Gharaibeh, M. *et al.* A look at router geolocation in public and commercial databases. In *Proceedings of the 2017 Internet Measurement Conference*, 463–469 (ACM, 2017).

[26] Shavitt, Y. & Zilberman, N. A geolocation databases study. *IEEE Journal on Selected Areas in Communications* **29**, 2044–2056 (2011).

[27] Poese, I., Uhlig, S., Kaafar, M. A., Donnet, B. & Gueye, B. Ip geolocation databases: Unreliable? *ACM SIGCOMM Computer Communication Review* **41**, 53–56 (2011).

[28] Laki, S. *et al.* Spotter: A model based active geolocation service. In *INFOCOM, 2011 Proceedings IEEE*, 3173–3181 (IEEE, 2011).

[29] Wong, B., Stoyanov, I. & Sirer, E. G. Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts. In *NSDI 2007*, 23–23 (USENIX Association, Berkeley, CA, USA, 2007). URL `http://dl.acm.org/citation.cfm?id=1973430.1973453`.

[30] Guo, C. *et al.* Mining the Web and the Internet for Accurate IP Address Geolocations. In *INFOCOM 2009*, 2841–2845 (2009).

[31] Chandrasekaran, B. *et al.* Alidade: IP geolocation without active probing. Tech. Rep., Department of Computer Science, Duke University (2015).

[32] Postel, J. *et al.* Rfc 792: Internet control message protocol. RFC 792, RFC Editor (1981). URL `https://tools.ietf.org/html/rfc792`.

[33] Hawkinson, J. & Bates, T. Guidelines for creation, selection, and registration of an autonomous system (as) (1996). URL `https://tools.ietf.org/html/rfc1930`.

[34] Rekhter, Y., Li, T. & Hares, S. A border gateway protocol 4 (bgp-4) (2006). URL `https://tools.ietf.org/html/rfc4271`.

[35] Yi, X., Raghavan, H. & Leggetter, C. Discovering users' specific geo intention in web search. In *Proceedings of the 18th international conference on World wide web*, 481–490 (ACM, 2009).

[36] Lu, Y., Peng, F., Wei, X. & Dumoulin, B. Personalize web search results with user's location. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 763–764 (ACM, 2010).

[37] Hoos, H. H. & Stützle, T. *Stochastic local search: Foundations and applications* (Elsevier, 2004).

[38] Hofmann-Wellenhof, B., Lichtenegger, H. & Collins, J. *Global Positioning System: Theory and Practice* (Springer, 1993).

[39] Gueye, B., Ziviani, A., Crovella, M. & Fdida, S. Constraint-Based Geolocation of Internet Hosts. *IEEE/ACM Transactions on Networking* **14**, 1219–1232 (2006).

[40] Mills, D. Internet delay experiments. RFC 792, RFC Editor (1983). URL `https://tools.ietf.org/html/rfc889`.

[41] Berry, B. J. L. City Size Distributions and Economic Development. *Economic Development and Cultural Change* **9**, 573–588 (1961). URL `http://www.jstor.org/stable/1151867`.

[42] Han, J., Pei, J. & Kamber, M. *Data mining: concepts and techniques* (Elsevier, 2011).

[43] Malkin, G. S. Traceroute using an ip option (1993). URL `https://tools.ietf.org/html/rfc1393`.

[44] Bing Geocoding API. `http://msdn.microsoft.com/en-us/library/ff701711.aspx` ((accessed July 17, 2015)).

[45] Fox, S., Karnawat, K., Mydland, M., Dumais, S. & White, T. Evaluating Implicit Measures to Improve Web Search. *ACM Transactions on Information Systems* **23**, 147–168 (2005). URL `http://doi.acm.org/10.1145/1059981.1059982`.

[46] El-Rabbany, A. *Introduction to GPS: The Global Positioning System*. Artech House mobile communications series (Artech House, 2002). URL `https://books.google.com/books?id=U2JmghrrB8cC`.

[47] Hofmann-Wellenhof, B., Lichtenegger, H. & Wasle, E. *GNSS – Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more* (Springer, 2007).

[48] Cheng, Y.-C., Chawathe, Y., LaMarca, A. & Krumm, J. Accuracy characterization for metropolitan-scale wi-fi localization. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, 233–245 (ACM, 2005).

[49] EdgeScape, A. Identify the geographic location and network characteristics of your users (2019). URL https://developer.akamai.com/edgescape. Accessed: 2019-02-05.

[50] IPligence. IP geo-location solutions for the internet (2019). URL https://www.ipligence.com/. Accessed: 2019-01-23.

[51] Digital Element. Geolocation and IP Intelligence Leader (2019). URL https://www.digitalelement.com/. Accessed: 2019-01-23.

[52] Digital Element. Geolocation and IP Intelligence Leader (2019). URL https://www.digitalelement.com/resources/faq/. Accessed: 2019-02-06.

[53] Digital Element. NetAcuity Pulse (2019). URL https://www.digitalelement.com/solutions/netacuity-pulse/. Accessed: 2019-02-06.

[54] Parekh, S. M., Friedman, R. B., Tibrewala, N. K. & Lutch, B. Systems and methods for determining collecting and using geographic locations of internet users (2004). US Patent 6,757,740.

[55] Parekh, S. Determining geographic locations of private network internet users (2006). US Patent App. 11/233,087.

[56] Rizzuto, J. J., Burdette, J. & Nystrom, J. Methods and systems for determining reverse dns entries (2010). US Patent 7,808,925.

[57] Burdette, J. *et al.* Method, computer program product and electronic device for hyper-local geo-targeting (2013). US Patent 8,443,107.

[58] Friedman, R. How does IP geolocation service providers collect data or how does IP geolocation databases are filled? (2012). URL https://www.quora.com/How-does-IP-geolocation-service-providers-collect-data-or-how-does-IP-geolocation-databases-are-filled. Accessed: 2019-02-06.

[59] Lee, Y., Park, H. & Lee, Y. Ip geolocation with a crowd-sourcing broadband performance tool. *ACM SIGCOMM Computer Communication Review* **46**, 12–20 (2016).

[60] IP2Location. Study of IPv4 Address Allocation by Continents (2018). URL `https://blog.ip2location.com/knowledge-base/study-of-ipv4-address-allocation-by-continents/`. Accessed: 2019-02-06.

[61] Neustar. The Proprietary Science Behind IP Intelligence (2019). URL `https://www.security.neustar/digital-performance/ip-intelligence/custom-geopoint`. Accessed: 2019-02-06.

[62] Liu, H. *et al.* Mining checkins from location-sharing services for client-independent ip geolocation. In *INFOCOM, 2014 Proceedings IEEE*, 619–627 (IEEE, 2014).

[63] Hubbard, K., Kosters, M., Conrad, D., Karrenberg, D. & Postel, J. Internet Registry IP Allocation Guidelines. Tech. Rep., RFC Editor, United States (1996).

[64] MaxMind. GeoIP2 City Accuracy (2019). URL `https://www.maxmind.com/en/geoip2-city-accuracy-comparison`. Accessed: 2019-03-18.

[65] IP2Location. Data Accuracy (2019). URL `https://www.ip2location.com/data-accuracy`. Accessed: 2019-03-18.

[66] Huffaker, B., Dhamdhere, A., Fomenkov, M. *et al.* Toward topology dualism: improving the accuracy of as annotations for routers. In *International Conference on Passive and Active Network Measurement*, 101–110 (Springer, 2010).

[67] CAIDA, A. Archipelago measurement infrastructure (2015).

[68] Staff, R. Ripe atlas: A global internet measurement network. *Internet Protocol Journal* **18** (2015).

[69] Bing Reverse Geocoding API. `http://msdn.microsoft.com/en-us/library/ff701710.aspx` ((accessed July 17, 2015)).

[70] Padmanabhan, V. N. & Subramanian, L. Determining the geographic location of internet hosts. In *SIGMETRICS/Performance*, 324–325 (2001).

[71] Padmanabhan, V. N. & Subramanian, L. An Investigation of Geographic Mapping Techniques for Internet Hosts. In *SIGCOMM 2001*, 173–185 (ACM, San Diego, California, USA, 2001). URL `http://doi.acm.org/10.1145/383059.383073`.

[72] Ziviani, A., Fdida, S., De Rezende, J. F. & Duarte, O. C. M. Toward a measurement-based geographic location service. In *International Workshop on Passive and Active Network Measurement*, 43–52 (Springer, 2004).

[73] Gueye, B., Ziviani, A., Crovella, M. & Fdida, S. Constraint-based geolocation of internet hosts. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 288–293 (ACM, 2004).

[74] Youn, I., Mark, B. & Richards, D. Statistical Geolocation of Internet Hosts. In *ICCCN 2009*, 1–6 (2009).

[75] Chun, B. *et al.* Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review* **33**, 3–12 (2003).

[76] Ferguson, A. D., Place, J. & Fonseca, R. Growth analysis of a large isp. In *Proceedings of the 2013 conference on Internet measurement conference*, 347–352 (ACM, 2013).

[77] Dong, Z., Perera, R. D., Chandramouli, R. & Subbalakshmi, K. Network measurement based modeling and optimization for ip geolocation. *Computer Networks* **56**, 85–98 (2012).

[78] Khan, R. A. *et al.* Adaptive geolocation of internet hosts. Tech. Rep., SLAC National Accelerator Lab., Menlo Park, CA (United States) (2016).

[79] Jiang, H., Liu, Y. & Matthews, J. N. Ip geolocation estimation using neural networks with stable landmarks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*, 170–175 (IEEE, 2016).

[80] Ciavarrini, G., Greco, M. S. & Vecchio, A. Geolocation of internet hosts: Accuracy limits through cramér–rao lower bound. *Computer Networks* **135**, 70–80 (2018).

[81] Yang, B. & Scheuing, J. Cramer-rao bound and optimum sensor array for source localization from time differences of arrival. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, vol. 4, iv–961 (IEEE, 2005).

[82] Qi, Y., Kobayashi, H. & Suda, H. Analysis of wireless geolocation in a non-line-of-sight environment. *IEEE Transactions on wireless communications* **5**, 672–681 (2006).

[83] Zheng, X. *et al.* A study of localization accuracy using multiple frequencies and powers. *IEEE Transactions on Parallel and Distributed Systems* **25**, 1955–1965 (2014).

[84] Matthews, W. & Cottrell, L. The pinger project: active internet performance monitoring for the henp community. *IEEE Communications Magazine* **38**, 130–136 (2000).

[85] Jayant, R. & Katz-Bassett, E. Toward better geolocation: Improving internet distance estimates using route traces. Tech. Rep., The Pennsylvania State University (2004).

[86] Katz-Bassett, E. *et al.* Towards ip geolocation using delay and topology measurements. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 71–84 (ACM, 2006).

[87] Spring, N., Mahajan, R. & Wetherall, D. Measuring isp topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review* **32**, 133–145 (2002).

[88] Bureau, U. C. Tiger/line shapefiles. *US Census Bureau* (2010).

[89] Hijmans, R., Garcia, N. & Wieczorek, J. Gadm: database of global administrative areas. *V ersion* (2010).

[90] Dovrolis, C., Gummadi, K., Kuzmanovic, A. & Meinrath, S. D. Measurement lab: Overview and an invitation to the research community. *ACM SIGCOMM Computer Communication Review* **40**, 53–56 (2010).

[91] Laki, S., Mátray, P., Hága, P., Csabai, I. & Vattay, G. A model based approach for improving router geolocation. *Computer Networks* **54**, 1490–1501 (2010).

[92] Gueye, B., Uhlig, S., Ziviani, A. & Fdida, S. Leveraging buffering delay estimation for geolocation of internet hosts. In *International Conference on Research in Networking*, 319–330 (Springer, 2006).

[93] Eriksson, B., Barford, P., Sommers, J. & Nowak, R. A learning-based approach for ip geolocation. In *International Conference on Passive and Active Network Measurement*, 171–180 (Springer, 2010).

[94] Eriksson, B., Barford, P., Maggs, B. & Nowak, R. Posit: An adaptive framework for lightweight ip geolocation. Tech. Rep., Computer Science Department, Boston University (2011).

[95] Eriksson, B., Barford, P., Maggs, B. & Nowak, R. Posit: a lightweight approach for ip geolocation. *ACM SIGMETRICS Performance Evaluation Review* **40**, 2–11 (2012).

[96] Chabarek, J. & Barford, P. What's in a name?: decoding router interface names. In *Proceedings of the 5th ACM workshop on HotPlanet*, 3–8 (ACM, 2013).

[97] Freedman, M. J., Vutukuru, M., Feamster, N. & Balakrishnan, H. Geographic locality of ip prefixes. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, 13–13 (USENIX Association, 2005).

[98] Wei, L., Ren, G., Shi, L., Tao, Y. & Cao, Y. 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*, 1060–1064 (IEEE, 2013).

[99] NUR, A. Y. & TOZAL, M. E. Geography and Routing in the Internet. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* (2018).

[100] Huffaker, B., Fomenkov, M. & claffy, k. DRoP:DNS-based Router Positioning. *ACM SIGCOMM Computer Communication Review (CCR)* **44**, 6–13 (2014).

[101] for Applied Internet Data Analysis, C. DDec - DNS Decoded - CAIDA's public DNS Decoding database (2018). URL `http://ddec.caida.org/help.pl`. Accessed: 2018-07-31.

[102] Scheitle, Q., Gasser, O., Sattler, P. & Carle, G. Hloc: Hints-based geolocation leveraging multiple measurement frameworks. In *Network Traffic Measurement and Analysis Conference (TMA), 2017*, 1–9 (IEEE, 2017).

[103] Wang, Y., Burgener, D., Flores, M., Kuzmanovic, A. & Huang, C. Towards Street-level Client-independent IP Geolocation. In *NSDI 2011*, 365–379 (USENIX, Berkeley, CA, USA, 2011). URL `http://dl.acm.org/citation.cfm?id=1972457.1972494`.

[104] Backstrom, L., Sun, E. & Marlow, C. Find Me if You Can: Improving Geographical Prediction with Social and Spatial Proximity. In *WWW 2010*, 61–70 (ACM, Raleigh, North Carolina, USA, 2010). URL `http://doi.acm.org/10.1145/1772690.1772698`.

[105] Cheng, Z., Caverlee, J. & Lee, K. You are where you tweet: a content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, 759–768 (ACM, 2010).

[106] Davis Jr, C. A., Pappa, G. L., de Oliveira, D. R. R. & de L. Arcanjo, F. Inferring the location of twitter messages based on user relationships. *Transactions in GIS* **15**, 735–751 (2011).

[107] Chandra, S., Khan, L. & Muhaya, F. B. Estimating twitter user location using social interactions–a content based approach. In *Privacy, Security,*

Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on, 838–843 (IEEE, 2011).

[108] Li, R., Wang, S., Deng, H., Wang, R. & Chang, K. C.-C. Towards social user profiling: unified and discriminative influence model for inferring home locations. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1023–1031 (ACM, 2012).

[109] Chang, H.-w., Lee, D., Eltaher, M. & Lee, J. @ phillies tweeting from philly? predicting twitter user locations with spatial word usage. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, 111–118 (IEEE Computer Society, 2012).

[110] Jurgens, D. That's what friends are for: Inferring location in online social media platforms based on social relationships. *Icwsm* **13**, 273–282 (2013).

[111] Han, B., Cook, P. & Baldwin, T. Text-based twitter user geolocation prediction. *Journal of Artificial Intelligence Research* **49**, 451–500 (2014).

[112] Mahmud, J., Nichols, J. & Drews, C. Home location identification of twitter users. *ACM Transactions on Intelligent Systems and Technology (TIST)* **5**, 47 (2014).

[113] Hulden, M., Silfverberg, M. & Francom, J. Kernel density estimation for text-based geolocation. In *AAAI*, 145–150 (2015).

[114] Zheng, X., Han, J. & Sun, A. A survey of location prediction on twitter. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[115] Moore, D., Periakaruppan, R., Donohoe, J. & Claffy, K. Where in the world is netgeo. caida. org. In *INET 2000* (2000).

[116] Endo, P. & Sadok, D. Whois Based Geolocation: A Strategy to Geolocate Internet Hosts. In *AINA 2010*, 408–413 (2010).

[117] Li, D. *et al.* Ip-geolocation mapping for moderately connected internet regions. *IEEE Transactions on Parallel and Distributed Systems* **24**, 381–391 (2013).

[118] National Information Society Agency (NIA), K. Speed Broadband Performance Tool (2019). URL `http://speed.nia.or.kr/index.asp`. Accessed: 2019-02-02.

[119] Hussain, F. *et al.* Evaluation of ip geolocation algorithms on pinger and planet-lab infrastructures (2010).

[120] Dahnert, A. Hawkeyes: An advanced ip geolocation approach: Ip geolocation using semantic and measurement based techniques. In *Cybersecurity Summit (WCS), 2011 Second Worldwide*, 1–3 (IEEE, 2011).

[121] Hillmann, P., Stiemert, L., Rodosek, G. D. & Rose, O. Modelling of ip geolocation by use of latency measurements. In *Network and Service Management (CNSM), 2015 11th International Conference on*, 173–177 (IEEE, 2015).

[122] Hillmann, P., Stiemert, L., Rodosek, G. D. & Rose, O. Dragoon: advanced modelling of ip geolocation by use of latency measurements. In *Internet Technology and Secured Transactions (ICITST), 2015 10th International Conference for*, 438–445 (IEEE, 2015).

[123] Hillmann, P., Stiemert, L., Dreo, G. & Rose, O. On the path to high precise ip geolocation: A self-optimizing model. *International Journal of Intelligent Computing Research (IJICR)* (2016).

[124] Koch, R., Golling, M. & Rodosek, G. D. Advanced geolocation of ip addresses. In *International Conference on Communication and Network Security (ICCNS)*, 1–10 (2013).

[125] Hyun, Y. *et al.* The caida ipv4 routed/24 topology dataset. `http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml`. Accessed: 2019-01-02.

[126] McGregor, T., Braun, H.-W. & Brown, J. The nlanr network analysis infrastructure. *IEEE Communications Magazine* **38**, 122–128 (2000).

[127] Hanemann, A. *et al.* Perfsonar: A service oriented architecture for multi-domain network monitoring. In *International conference on service-oriented computing*, 241–254 (Springer, 2005).

[128] Spring, N. T., Wetherall, D. & Anderson, T. E. Scriptroute: A public internet measurement facility. In *USENIX Symposium on Internet Technologies and Systems* (2003).

[129] Rainie, L. & Duggan, M. Privacy and information sharing. *Pew Research Center* **16** (2016).

[130] Center, P. R. Location-Based Services (2013). URL `http://www.pewinternet.org/2013/09/12/location-based-services/`. Accessed: 2019-02-06.

[131] Valentino-DeVries, J., Singer, N., Keller, M. H. & Krolik, A. Your apps know where you were last night, and they're not keeping it secret. *New York Times, Dec* **10** (2018). URL `https://www.nytimes.com/interactive/2018/12/10/business/location-data-privacy-apps.html`. Accessed: 2019-02-06.

[132] Times, T. N. Y. How the times analyzed location tracking companies. *New York Times, Dec* **10** (2018). URL `https://www.nytimes.com/2018/12/10/technology/location-tracking-apps-privacy.html`. Accessed: 2019-02-06.

[133] Mobile, R. Win More Business with Location-Based Marketing & Analytics (2019). URL `https://revealmobile.com/`. Accessed: 2019-02-06.

[134] SafeGraph. The Source of Truth for Physical Places (2019). URL `https://www.safegraph.com/`. Accessed: 2019-02-06.

[135] Kiip. Moments Based In-App Mobile Advertising (2019). URL `http://www.kiip.me/`. Accessed: 2019-02-06.

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

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

[138] Muir, J. A. & Oorschot, P. C. V. Internet geolocation: Evasion and counterevasion. *ACM Computing Surveys (CSUR)* **42**, 4 (2009).

[139] Gill, P., Ganjali, Y., Wong, B. & Lie, D. Dude, where's that ip?: circumventing measurement-based ip geolocation. In *Proceedings of the 19th USENIX conference on Security*, 16–16 (USENIX Association, 2010).

[140] Abdou, A., Matrawy, A. & van Oorschot, P. C. Accurate manipulation of delay-based internet geolocation. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 887–898 (ACM, 2017).

[141] Chaiken, R. *et al.* Scope: Easy and efficient parallel processing of massive data sets. *Proc. VLDB Endow.* **1**, 1265–1276 (2008). URL `http://dx.doi.org/10.14778/1454159.1454166`.

[142] Wang, L. *et al.* Detecting Dominant Locations from Search Queries. In *SIGIR 2015*, 424–431 (ACM, Salvador, Brazil, 2005). URL `http://doi.acm.org/10.1145/1076034.1076107`.

[143] Yahoo! PlaceSpotter. `https://developer.yahoo.com/boss/geo/docs/key-concepts.html` ((accessed July 17, 2015)).

[144] OpenCalais. `http://www.opencalais.com/` ((accessed July 17, 2015)).

[145] Finkel, J. R., Grenager, T. & Manning, C. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *ACL 2005*, 363–370 (Ann Arbor, Michigan, 2005). URL `http://dx.doi.org/10.3115/1219840.1219885`.

[146] Cunningham, H., Maynard, D., Bontcheva, K. & Tablan, V. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *ACL 2002*, 168–175 (2002).

[147] Google Geocoding API. https://developers.google.com/maps/documentation/geocoding/ ((accessed July 17, 2015)).

[148] White, T. *Hadoop: The Definitive Guide*. O'Reilly and Associates Series (O'Reilly, 2012). URL https://books.google.com/books?id=drbI_aro20oC.

[149] Mockapetris, P. DOMAIN NAMES - CONCEPTS AND FACILITIES. RFC 1034, RFC Editor (1987). URL https://tools.ietf.org/html/rfc1034.

[150] Eidnes, H., de Groot, G. & Vixie, P. Classless IN-ADDR.ARPA delegation. RFC 2317, RFC Editor (1998). URL https://tools.ietf.org/html/rfc2317.

[151] Wick, M. Geonames (2018). URL http://download.geonames.org/export/dump/. Accessed: 2018-06-27.

[152] Timmins, P. Telcodata telecommunications database (2018). URL https://www.telcodata.us/. Accessed: 2018-06-27.

[153] for Europe, U. N. E. C. UN/LOCODE: United Nations Code for Trade and Transport Locations (2018). URL https://www.unece.org/cefact/locode/welcome.html. Accessed: 2018-06-27.

[154] Foundation, M. Public suffix list (2018). URL https://publicsuffix.org/list/. Accessed: 2018-06-28.

[155] Rapid7Labs. Reverse dns (rdns) v2 - 2017 onward. Tech. Rep., Rapid7Labs (2018). URL https://opendata.rapid7.com/sonar.rdns_v2/. Accessed: 2018-06-23.

[156] Rapid7Labs. Reverse dns (rdns) - 2013-2017. https://opendata.rapid7.com/sonar.rdns/ (2017). Accessed: 2018-06-23.

[157] Internet Assigned Numbers Authority. IANA IPv4 Special-Purpose Address Registry. Tech. Rep., IANA (2017). URL https://www.iana.org/assignments/iana-ipv4-special-

registry/iana-ipv4-special-registry.xhtml. Accessed: 2018-08-10.

[158] Braden, R. Requirements for Internet Hosts – Application and Support. RFC 1123, RFC Editor (1989). URL https://tools.ietf.org/html/rfc1123.

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

[160] Misra, P. & Enge, P. Global positioning system: signals, measurements and performance second edition. *Massachusetts: Ganga-Jamuna Press* (2006).

[161] Ester, M., Kriegel, H.-P., Sander, J., Xu, X. *et al.* A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, vol. 96, 226–231 (1996).

[162] Amitay, E., Har'El, N., Sivan, R. & Soffer, A. Web-a-where: geotagging web content. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 273–280 (ACM, 2004).

[163] Silva, M. J., Martins, B., Chaves, M., Afonso, A. P. & Cardoso, N. Adding geographic scopes to web resources. *Computers, Environment and Urban Systems* **30**, 378–399 (2006).

[164] Martins, B., Anastácio, I. & Calado, P. A machine learning approach for resolving place references in text. In *Geospatial thinking*, 221–236 (Springer, 2010).

[165] Andrade, L. & Silva, M. J. Relevance ranking for geographic ir. In *GIR* (2006).

[166] Martins, B. & Calado, P. Learning to rank for geographic information retrieval. In *proceedings of the 6th workshop on geographic information retrieval*, 21 (ACM, 2010).

[167] Niemeyer, G. Geohash (2008).

[168] Zhang, S., Han, J., Liu, Z., Wang, K. & Feng, S. Spatial queries evaluation with mapreduce. In *Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on*, 287–292 (IEEE, 2009).

[169] Dan, O., Parikh, V. & Davison, B. D. Improving ip geolocation using query logs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 347–356 (ACM, 2016).

[170] Gueye, B., Uhlig, S. & Fdida, S. Investigating the imprecision of ip block-based geolocation. In *International Conference on Passive and Active Network Measurement*, 237–240 (Springer, 2007).

[171] Lodhi, A., Larson, N., Dhamdhere, A., Dovrolis, C. *et al.* Using peeringdb to understand the peering ecosystem. *ACM SIGCOMM Computer Communication Review* **44**, 20–27 (2014).

[172] Trust, I. C. CAIDA UCSD IPv4 Prefix-Probing Traceroute Dataset (2019). URL `https://www.impactcybertrust.org/dataset_view?idDataset=720`. Accessed: 2019-04-09.

[173] ICANNWiki. Jon Postel (2019). URL `https://icannwiki.org/Jon_Postel`. Accessed: 2019-02-24.

[174] APNIC. History of the Regional Internet Registries (2019). URL `https://www.apnic.net/about-apnic/organization/history-of-apnic/history-of-the-regional-internet-registries/`. Accessed: 2019-02-24.

[175] APNIC. History of the Internet (2019). URL `https://www.apnic.net/about-apnic/organization/history-of-apnic/history-of-the-internet/#before-the-rirs`. Accessed: 2019-02-24.

[176] Housley, R., Curran, J., Huston, G. & Conrad, D. Rfc 7020: The internet numbers registry system. RFC 7020, RFC Editor (2013). URL `https://tools.ietf.org/html/rfc7020`.

[177] Williamson, S., Kosters, M., Blacka, D., Singh, J. & Zeilstra, K. Referral whois (rwhois) protocol v1.5 (1997). URL `https://tools.ietf.org/html/rfc2167`.

[178] ARIN. ARIN Number Resource Policy Manual (2019). URL `https://www.arin.net/policy/nrpm.html#three2`. Accessed: 2019-02-24.

[179] WEIRDS, I. Unified Object Inventory (2013). URL `https://www.ietf.org/proceedings/88/slides/slides-88-weirds-2.pdf`. Accessed: 2019-02-24.

[180] John, G. H. & Langley, P. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 338–345 (Morgan Kaufmann Publishers Inc., 1995).

[181] Le Cessie, S. & Van Houwelingen, J. C. Ridge estimators in logistic regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **41**, 191–201 (1992).

[182] Quinlan, J. R. *C4. 5: programs for machine learning* (Elsevier, 2014).

[183] Kohavi, R. The power of decision tables. In *European conference on machine learning*, 174–189 (Springer, 1995).

[184] Platt, J. C. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning* (MIT Press, 1998). URL `https://www.microsoft.com/en-us/research/publication/fast-training-of-support-vector-machines-using-sequential-minimal-optimization/`.

[185] Freund, Y., Schapire, R. E. *et al.* Experiments with a new boosting algorithm. In *icml*, vol. 96, 148–156 (Citeseer, 1996).

[186] Rodriguez, J. J., Kuncheva, L. I. & Alonso, C. J. Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence* **28**, 1619–1630 (2006).

Vita

Ovidiu Dan was born in Târgu Mureş, Romania, on February 8th, 1986 to Otilia Dan and Felician Dan. He graduated from National College "Alexandru Papiu Ilarian" Târgu Mureş in 2005. Between 2005 and 2009 he studied Information Technology at Inholland University of Applied Sciences, located in Amsterdam, The Netherlands, where he received his Bachelor of Engineering. As part of his PhD studies, he received a Master of Science in Computer Science from Lehigh University in 2012.