Mining the Web and the Internet for Accurate IP Address Geolocations

Chuanxiong Guo, Yunxin Liu, Wenchao Shen†, Helen J. Wang, Qing Yu, Yongguang Zhang
{chguo, yunliu, helenw, qingyu, ygz}@microsoft.com, †shenwenchao@gmail.com
Microsoft Research, †Tsinghua University

***Abstract*— In this paper, we present Structon, a novel approach that uses Web mining together with inference and IP traceroute to geolocate IP addresses with significantly better accuracy than existing automated approaches. Structon is composed of three ideas which we realize in three corresponding steps. First, we extract geolocation information of Web server IP addresses from Web pages. Second, we devise heuristic algorithms to improve both the accuracy and the coverage of the IP geolocation database using these Web server IP addresses and their geolocations as input. Third, for those segments that are not covered in the first two steps, we use IP traceroute to identify the access routers of those segments. When the location of the access router is known, we can deduce the location of the associated segment since it is co-located together with the access router.**

By mining 500-million Web pages collected in China in 2006 (11 percent of the total Web pages in China at that time), we are able to identify the geolocations for 103 million IP addresses. This represents nearly 88 percent IP addresses allocated to China in March 2008. Structon is 87.4 percent accurate at city granularity and up to 93.5 percent accurate at province level. We also used 10 day Windows Live client log to evaluate our client IP addresses coverage: Structon identified geolocations of 98.9 percent of client IP addresses.

I. Introduction

The geographical location (i.e., geolocation) of an IP address is important for many location-aware Internet applications, such as online advertisement targeting, extracting customer geolocation distribution from Web log analysis, and locality-aware P2P overlay construction.

In this paper, we propose a novel approach, which we call Structon, for accurate IP address to geolocation mapping. The key observation of Structon is that Web content embeds rich geolocation information, such as province (state), city, zipcode, and telephone area code, which can be leveraged to map the Web server's IP addresses. With this initial mapping, we can design inference algorithms to improve both coverage and accuracy. And we can further leverage other information, such as IP segments geolocation distribution, AS (autonomous systems) information from WHOIS database, and BGP routing table, and path information gotten from traceroute measurement to infer the geolocations of additional IP addresses including client IP addresses as well as to correct errors in the initial mapping. Structon is composed of three parts which we briefly describe as follows.

In the first part, we extract geolocations from Web pages with accurate pattern matching and get the possible locations of Web servers by information clustering. In the second part, we design a series of multi-stage inferring heuristics to both increase the coverage and accuracy ratio of our IP to geolocation database. In this part, we use majority voting and leverage information from BGP and WHOIS databases to increase coverage and improve accuracy. In the third part, we use IP traceroute to further increase coverage of our database.

By mining 500-million Web pages collected in China in 2006 (which corresponds to 11 percent of the total Web pages in China at that time), we observe that 66 percent of Web server IP addresses can be correctly mapped to correct cities, and 73 percent mapped to correct provinces. Our inference algorithm significantly improves the accuracy to about 87.4 percent at the city level, and 93.5 percent at province level. To understand the client IP address coverage of Structon, we analyzed on a 10 day-long Windows Live client log, which consists of 8.5 million unique IP addresses. We found that Structon can cover more than 98.9 percent of these IP addresses.

Compared with other IP geolocation solutions such as delay-based [1], [5], [7] and data-mining-based [6], [7], and the commercial IP geolocation databases [8], [3], Structon provides a new solution based on Web mining, which is more accurate.

II. Geolocation extraction and clustering

In geolocation extraction and clustering, our purpose is to identify the possible geolocations of Web server IP addresses by extracting city and province (state) names, zipcodes, telephone area numbers from Web pages. To do so, we carry out the following two steps sequentially: In the first step, we extract geolocation information from every Web page using regular expressions for location pattern matching. In the second step, we cluster the geolocation information of the same DNS (domain name system) name together to form a location weight vector (LWV) for that DNS name. This location weight vector contains all the possible locations of the Web server and is then assigned to the corresponding IP addresses.

A. Geolocation extraction

The geolocation information we care is the contact information of the organization that owns the Web site. The contact information includes city and province names, telephone area numbers, and zipcodes, which can be collected from the Internet or bought from commercial products.

url \ location	Loc_a	Loc_b	Loc_c	Loc_d
dns_a/url1	0.64	-	0.96	0.89
dns_a/url2	0.64	-	0.95	0.89
dns_a/url3	-	0.57	0.95	0.86
LWV of dns_a	0.43	0.19	0.95	0.88

TABLE I

CALCULATING THE LOCATION WEIGHT VECTOR (LWV) OF A DNS NAME. THE URLS IN THE TABLE SHARE THE SAME DNS NAME.

IP\ location	Loc_a	Loc_b	Loc_c	Loc_d	Loc_e
61.155.111.42	0.003	0.004	0.003	0.24	-
61.155.111.44	-	0.02	-	-	-
61.155.111.70	-	0.77	-	-	0.13
Location PDF	0.26%	68%	0.26%	20.5%	11%

TABLE II

CALCULATE THE LOCATION PROBABILITY DISTRIBUTION FUNCTION (PDF) OF A /24 SEGMENT FROM THE LOCATION WEIGHT VECTORS OF THE IP ADDRESSES IN THAT SEGMENT.

We first parse each HTML file into a list of chunks based on the HTML tags. Each chunk is roughly a visible line of string in the Web page. For each chunk, we use regular expression for geolocation extraction.

When we successfully extract a geolocation item, we assign it a weight. Currently the weight is assigned based on its position in the page. We observe that one item is more likely to be a contact information if it appears in the bottom of a page (this is true for Chinese Web pages; for Web pages in the US and the western countries, contact information may appear in the beginning of a Web page). In this paper, we assign weight to an item in proportion to its position. For each Web page, the output of the extraction algorithm is a list of geolocation items and their corresponding weights.

Our extraction algorithm is tuned to be executed very fast. We can finish extraction for 500-million Web pages in 18 hours using a 50-machines cluster. The processing time for each page is about 6 milliseconds. This 6 milliseconds including both HTML parsing and regular expression matching.

B. IP address to geolocation mapping

We cluster the Web pages of the same DNS name together and calculate a location weight vector (LWV) for that DNS name. The procedure can be explained using the example illustrated in Table I.

In this example, *dns_a* has three urls, url1, url2, and url3. Each url contains several geolocation items with different weights. We calculate the mean weight for each gelocation. For example, in Table I, the mean weight for Loc_a is therefore $(0.64+0.64)/3 \approx 0.43$. Similarly, the weights for Loc_b, Loc_c, and Loc_d are 0.19, 0.95, 0.88, respectively. We then use these mean values to form a LWV for *dns_a*: {Loc_a:0.43, Loc_b:0.19, Loc_c:0.95, Loc_d:0.88}.

For each {dns, LWV} pair, we resolve the DNS name to IP addresses. One DNS name may be mapped to multiple IP addresses. Each IP address will get a copy of the LWV of the corresponding DNS name. Multiple DNS names may be resolved to the same IP address. One IP address therefore may get several different LWVs from different DNS names. In this case, we normalize the multiple LWVs into a new LWV for that IP address.

III. MULTI-STAGE INFERENCE

In this paper, we assume that the IP addresses in the same /24 segments are in the same city since we have the tradition to use /24 as the smallest segment allocation granularity.

In multi-stage inference, the input is a set of Web server IP addresses and their location weight vectors produced in Section II. Our purpose is to increase both coverage and accuracy. One observation we made is that: Network administrators tend to allocate continuous IP segments to the same location. This simplifies the allocation procedure, and most importantly, significantly reduces the size of IP routing tables. The principle we use in the inferring heuristic algorithms therefore is majority voting: if most subsegments in a large segment say that they are in the same location, we infer that the large segment is in that location.

Our multi-stage inferring has three stages which we present sequentially.

A. Stage I: Location calculation for /24 segments

In order to decide the location of a /24 segment, we calculate the location probability distribution function (PDF) of that segment from the LWVs of the IP addresses in that segment. We then consider that the location of that segment is the location with the highest probability in the distribution.

The PDF calculation can be illustrated in the example in Table II. We first get the total sum of the weights in the table and the weight sum of each location (i.e., each column in the table). The probability of a location is then the weight sum of the location divided by the total sum.

As to the example in Table II, the total sum is 1.17, and the probabilities that the segment 61.155.111.0/24 is located in Loc_a, Loc_b, Loc_c, Loc_d, and Loc_e are 0.003/1.17, 0.794/1.17, 0.003/1.17, 0.24/1.17, and 0.13/1.17, respectively. Since Loc_b has the largest probability, we therefore conclude that this segment is in Loc_b.

As we will see in Section V-B, Stage I improves the accuracy by more than 10 percent at both city and province levels. Before Stage I, the accuracy is 70 percent at both province and city levels. After Stage I, the accuracy becomes more than 80 percent.

B. Stage II: Iterative inferring and error correction

In this subsection, we introduce two heuristics: inferring based on location pattern and error-correction based on majority voting. These two heuristics purely use the output of stage I and do not need any other data sources.

The structure of Stage II is depicted in Fig. 1. The output of the first heuristic is the input for the second heuristic. The whole Stage II procedure is an iterative one in that the output of the second heuristic is looped back to the first heuristic as

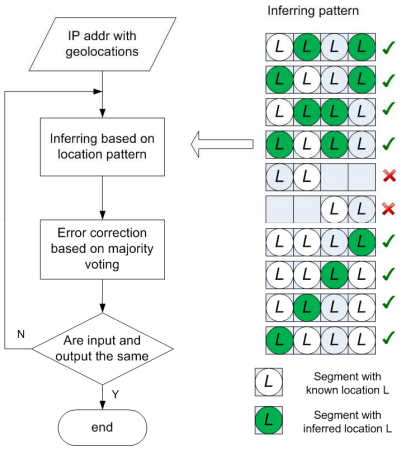

Fig. 1. Stage II inferring. It is composed of two heuristics: inferring based on location patterns and error-correction based on majority voting.

input. The iteration doesn't end until no further inferring can be performed in both heuristics.

1) Inferring based on location patterns: The location patterns we use are illustrated in Fig. 1. Using a /22 segment as an example, we number the four /24 sub-segments as $S_0 - S_3$ sequentially. We then study the location patterns of $S_0 - S_3$ and use the following rules to identify the location of the /22 segment:

- If the locations of the three of the four sub-segments are in the same location L, and the location of the rest one is undetermined, we infer that the whole /22 segment is in L;
- If the locations of two of the four sub-segments are in the same location L and locations of the other two sub-segments are unknown, we study how the two sub-segments with known location distribute. If the two sub-segments are clustered together, we do not make any inferring; otherwise, we deduce that the whole /22 segment is in L;

We perform the above inferring procedure by iteratively increasing the size of the segment. For example, we start from size /22 (which has four /24 subsegments), then we increase the size to /21, /20, etc. We denote the number of iterations as ni. The larger the number of iterations, the larger the coverage. The number of iterations, however, should not be too large. As the number of iterations increases, more errors will be introduced. In this paper, we set ni to 5, which means that we run the iterations for 5 times, from /22 to /18.

2) Error correction based on majority voting: Since Structon is a data mining approach, errors are inevitable. In what follows, we introduce a simple error correction heuristic based on majority voting. The idea is that if a majority portion of a segment agree that this segment is in a location L, whereas only a small number of subsegments say that the segment is in other different locations, we consider that the segment is in location L. We therefore (1) correct the 'errors' introduced by the small number of sub-segments; (2) infer the locations for the sub-segments that we previous do not know their locations.

The following notation is used to describe the heuristic: $IPSegList$ is the ascendant sorted list of the /24 segments that we know their locations in the same larger segment. $IPSeg_b$ and $IPSeg_e$ are the first and last Segments in $IPSegList$, N_a is the number of /24 segments in $IPSegList$. L_m is the location that the majority /24 segments in $IPSegList$ are located in. N_m is the number of /24 segments that are located in L_m.

Our error correction heuristic then works as follows. Segments in [$IPSeg_b$ - $IPSeg_e$] are considered to locate in L_m if the following three conditions are all met:

- The /24 sub-segments in $IPSegList$ are in the same larger segment;
- $N_a \geq nthresh_{22}$ and $\frac{N_m}{N_a} \geq mthresh_{22}$;
- Both $IPSeg_b$ and $IPSeg_e$ are in L_m.

In this paper, we set $nthresh_{22} = 10$ and $mthresh_{22} = 0.6$.

C. Stage III: Inferring by leveraging AS and BGP information

In Stage III, we further introduce two heuristics: inferring for small ISPs and Inferring by leveraging BGP routing table, to perform further location inferring. Both BGP routing table and AS information can be freely accessed from the Internet. The AS description can be accessed at the regional NIC (network information center) WHOIS database. These descriptions generally contain geolocation descriptions of the AS. And many research institutes such as [9] provide BGP routing table snapshots. In what follows, we describe these two heuristics in detail.

1) Inferring for small ASes: We observe from the BGP routing table that many AS numbers contain only a small number of IP addresses. These ASes are therefore small ISPs. These small ISPs are very likely to be located in a same province or city. If some of the IP segments of a small AS is in a location L, it is a good indication that the whole ISP is in L.

This type of inferring must be carried out very carefully with the following constrains:

1) The number of IP addresses in the AS should be smaller than a threshold $nthresh_{31}$. This threshold value is different for city level and province level inferring;
2) The number of IP addresses with known geolocations in the majority location L should be much larger than the number of IP addresses at other locations, i.e., the majority ratio should be larger than a threshold $mthresh_{31}$;
3) The deduced location L should be in coherent with the description of the AS in the WHOIS database.

In this paper, $nthresh_{31}$ is set to 2^{16} for city-level inferring and 2^{21} for province-level inferring, and $mthresh_{31} = 0.7$. See details on parameter setting in our technical report [2].

2) Inferring by leveraging BGP routing table: From the BGP routing table, we can get the original AS number of an IP segment. If two neighboring segments in BGP routing table are

```
/* We use the IP segments in BGP table as input */
for each segment S in BGP routing table:
    RecursiveInferring(S);
RecursiveInferring(S):
    if (S is smaller than sthresh_32):
        return;
    Calculate cratio and mratio for S;
    if (cratio ≥ cthresh_32) and (mratio ≥ mthresh_32):
        map S to L;
    else:
        split S to two equal sub-segments S_1 and S_2;
        RecursiveInferring(S_1);
        RecursiveInferring(S_2);
    return;
```

Fig. 2. Recursive Inferring using IP segments in BGP routing table as input.

in different ASes, they are very unlikely in the same location. Hence IP segments in BGP routing table provide us natural segmentation of the IP address space. We have designed an inferring heuristic, by recursively divide segments into sub-segments using IP segments in BGP routing table as input. Once the most of the IP addresses in that subsegment say that they are in the same location L, we then infer that these subsegments are in L. The heuristic is described in Fig. 2.

Suppose we denote the set of IP addresses with known locations as Set_k, the set of IP addresses at location L_m as Set_m, where L_m is the location with the largest number of IP addresses in Set_k. We define the coverage ratio $cratio = |Set_k|/|S|$ (where $|Set_k|$ and $|S|$ denote the number of IP addresses in Set_k and S, respectively) and majority ratio $mratio = |Set_m|/|Set_k|$.

The inferring criteria is that: we need both the coverage ratio and majority ratio of a segment to be larger than threshold, $cthresh_{32}$ and $mthresh_{32}$. That is, $cratio \geq cthresh_{32}$ and $mratio \geq mthresh_{32}$. In this paper, $cthresh_{32}$ is set to 0.1 and $mthresh_{32}$ is set to 0.9. See details on parameter setting in our technical report [2].

IV. Traceroute

In this section we introduce IP traceroute to further increase the coverage of our IP geolocation database. We assume that a /24 segment is located together with its access router R_n.

The idea works as follows. For each /24 segment in the BGP table, we use traceroute to identify a path from a source IP to a chosen IP address in that segment. Hence if the location of the access router R_n is known to be in Loc_a, we associate the segment with Loc_a. We call this forward inference. Similarly, if we know the location of the /24 segment, we can deduce the location of the access router and the rest of its associated /24 segments. We call this backward inference. It is apparent that forward inference increases the coverage ratio. Backward inference can also increases the coverage ratio. The reason is as follows. Suppose two segments A and B are connected to the Internet via access router R_n and we only know the location of A. By using backward inference, we get the location of R_n from A, we then further get the location of B from R_n using forward inference.

Number of Web pages	502,880,364
DNS names	3,991,164
Web server IP addr	231,501
Web pages with location information	124,143,191
DNS names with location information	549,437
Mined IP addr with location information	157,407
Deduced IP addr with location information	100,937,472

TABLE III
THE RESULTS WE GET FROM THE 500M WEB DATA ARCHIVE.

We applied our traceroute technique at three different sites in Beijing, Shanghai, and Guanzhou. The probing finished in five days. We then input these new segments together with the segments that we already know their locations into the inference heuristics. This significantly increases the coverage ratio as we will see in Section V. The traceroute and inference can be applied iteratively until the resulted IP geolocation database converges.

V. Experiment and Evaluation

A. Experiment

We have run Structon with a Web data archive that contains 500-millions Web pages located in China as input. It is collected by the Web Search and Mining Group (WSM) at Microsoft Research Asia (MSRA) in the end of 2006. The total number of Web pages in China was 4.5 billion at that time. The size of the Web data set is approximate 10TB. The dataset is stored in a distributed storage system. We use Dryad [4], which is a distributed execution engine, to extract geolocation information from every Web pages. The platform we use has 50 servers, each with Quad 2.2GHz AMD Opteron processors, 8GB memory, 1.4T hard disk, and 1Gb/s Ethernet network interface. The operating system is Windows Server 2003. It took 18 hours to finish the first step geolocation extraction using the 50-server cluster.

We then store the extracted {url, georecord} into a local server. The georecord includes the geolocations we get and their positions in the page, IP address from which the url is crawled, and the raw sentences that contain geolocations. All the rest computations (url clustering, Web server IP to location mapping, and multi-stage inference) are performed on that single server. In these computations, only the url clustering is CPU intensive and costs about 12 hours to complete. The rest procedures can all be finished in minutes, hence are not time critical.

Table III shows statistics of the Web data set and the final result we get.

B. Evaluation

1) Accuracy: Studying the accuracy of an IP geolocation scheme can be tricky since we do not have a complete groundtruth (otherwise we already have done). In this paper, we take the following approach to verify the accuracy of Structon: we use ip.cn, the grassroot generated IP geolocation database as our 'groundtruth'. ip.cn is a manually maintained database for IP addresses in China, which contains 120M IP

Database	total_ip	coherent_ratio/overlap_ip	
		province	city
WHOIS	93.8M	0.844/84.6M	0.488/62.6M
iputil_1	72.5M	0.759/65.8M	0.501/56.1M
iputil_2	125M	0.874/98.6M	0.684/72.5M
Structon	101M	0.935/89.0M	0.874/37.1M

TABLE IV

COMPARISON OF THE ACCURACIES OF STRUCTON WITH OTHER THREE IP GEOLOCATION DATABASES. *total_ip* IS THE NUMBER OF IPS IN THE DATABASE AND *overlap_ip* IS THE NUMBER OF IPS OVERLAPPED WITH IP.CN.

addresses. Our experiences showed that ip.cn is quite accurate, though it needs huge amount of human involvement (e.g., it costs about 6 years to establish such a database, and it needs end users' input to evolve).

We compare the accuracy of Structon at province and city levels with three geolocation databases: WHOIS, iputil_1, iputil_2. The WHOIS database is gotten by retrieving location information from the APNIC WHOIS source (whois.apnic.net). The technique we use is similar with that in [6]. iputil_1 and iputil_2 are commercial databases, which we got in 2007 and 2008, respectively. We do not reveal their product names due to privacy concerns.

The result is given in Table IV. The high coherent ratios between Structon and ip.cn indicates that both of them are of high accuracy. Therefore, our choosing of ip.cn as a 'truth' set is justified. Our result shows that Structon outperforms all the other three databases. The WHOIS database is very inaccurate at city level, which demonstrates that the information contained in WHOIS is either coarse-grained (e.g., the IP addresses of a large ISP may be registered to its headquarter) or wrong. Also the two commercial databases are less than 70 percent coherent with ip.cn at city level. This result is coherent with the accuracy number given in [3], which says that its accuracy at city level is 65 percent. The coherent ratio of Structon at province level is high and much better than those of the other three databases.

We also have studied the coherent ratios of Structon at different stages. These stages are: 'web mining', which we map Web server IP addresses to their geolocations; '+ /24 clustering', which we cluster the Web server IP addresses to /24 segments and get locations for the corresponding /24 segments; '+ inference', which we apply our inference heuristics; and '+ traceroute', which is the result we finally get. The result is given in Table V. It shows that clustering in the first stage and inference in the second stage improve both coverage and accuracy, whereas the third stage only increases coverage.

2) Coverage ratio: In this subsection, we study the IP address coverage of Structon. We first show that Structon covers a significant portion of IP addresses that appear in BGP routing table. We also show that Structon covers almost all the active IP addresses in practice.

Coverage for IP addresses in BGP: Using the 157,407 Web server IP addresses and their associated geolocations as input, we finally get geolocations for 101M IP addresses. This is 87.3 percent IP addresses in China in March 2008. (The RouteViews BGP routing table showed that China has 116M IP addresses in March 2008.)

	total_ip	coherent_ratio/overlap_ip	
		province	city
web mining	0.15M	0.730/0.11M	0.660/0.09M
+ /24 clustering	8.9M	0.850/8.8M	0.800/7.5M
+ inference	47.0M	0.947/46.2M	0.899/26.0M
+ traceroute	101M	0.935/89.0M	0.874/37.1M

TABLE V

THE IP ADDRESS NUMBERS GOTTEN AT DIFFERENT STAGES AND THEIR ACCURACY RATIOS AT PROVINCE AND CITY LEVELS.

Coverage for active IP addresses: We further use a large active IP address trace to study the coverage ratio for active IP addresses. The IP address trace is collected from the Windows Live search engine in China in ten days. Hence the IP trace collected is representative and it contains 8,500,189 unique IP addresses. The active IP addresses covered by Structon is 98.9 percent.

VI. CONCLUSION

In this paper, we have presented Structon, an automated approach that uses Web mining together with inference and IP traceroute, for accurate IP address to geolocation mapping. Using 500-million Web pages in China as input, Structon covers 87.3 percent IP addresses allocated to China and our client coverage study shows that the client IP coverage is 98.9 percent. Our study shows that the accuracy of Structon is more than 87 percent at city level and up to 93.5 percent at the province level.

VII. ACKNOWLEDGEMENT

We thank our students Chen Chen, Jiong Chen, Tao He for their work on this project. We thank Xiaohui Liu and Meijing Fang for providing us the client IP address log and Songwu Lu for valuable comments and suggestions.

REFERENCES

[1] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida. Constraint-Based Geolocation of Internet Hosts. *IEEE/ACM trans. Networking*, 14(6), Dec 2006.

[2] Chuanxiong Guo, Yunxin Liu, Wenchao Shen, Helen J. Wang, Qing Yu, and Yongguang Zhang. Mining the Web and the Internet for Accurate IP Address Geolocations, 2008. http://research.microsoft.com/users/chguo/structon.pdf.

[3] IP2LocationTM IP-Country-Region-City-ISP Database FAQ. http://www.ip2location.com/.

[4] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. In *Proc. ACM EuroSys'07*, Lisboa, Portugal, March 2007.

[5] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe. Towards IP Geolocation using Delay and Topology Measurements. In *Proceedings of IMC'06*, 2006.

[6] D. Moore, R. Periakaruppan, J. Donohoe, and k claffy. Where in the World is netgeo.caida.org? In *Proceedings of INET'00*, 2000.

[7] V. N. Padmanabhan and L. Subramanian. An Investigation of Geographic Mapping Techniques for Internet Hosts. In *Proceedings of SIGCOMM'01*, 2001.

[8] Quova: Location matters. http://www.quova.com.

[9] University of Oregon Route Views Project. http://www.routeviews.org/.