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 from Web pages and associate the extracted geolocations with the Web server IP addresses. 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. The second and third steps are run iteratively until the output converges.

By mining 500-million Web pages collected in China in 2006 (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.

1. 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. Structon takes Web pages as input and identifies the possible geolocations of Web server IP addresses by extracting and clustering geolocation information from Web pages. Structon then introduces inference heuristics to improve both accuracy and coverage of the IP geolocation database, by using the Web server IP addresses and the corresponding geolocations as input. These inference heuristics are based on the principle of majority voting and take advantage of the IP address segmentation property. 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. The inference and traceroute techniques can be used iteratively until the IP to geolocation database converges.

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 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.

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 [1]), 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 [8, 13, 14, 15, 18, 22, 23, 25] and data-mining-based [16, 18], and the commercial IP geolocation databases [19, 10], Structon provides a new solution...
based on Web mining, which is more accurate.

The organization of the paper is as follows. We discuss related work in Section 2. We describe the challenges faced by Structon and introduce the structure of Structon in Section 3. We describe how geolocation information are extracted and clustered in Section 4, how we perform inference in Section 5, and how we use IP traceroute measurement to further increase coverage in Section 6. We present experiments and evaluate the result in Section 7. Section 8 concludes the paper.

2. RELATED WORK

Work on IP gelocation can be classified into three different categories: data-mining-based, delay-based, and grassroot-based. Data-mining-based and delay-based approaches are automated approaches, whereas grassroot-based approach needs human input and is a manual approach.

2.1 Data-mining-based

Data-mining-based approaches get geolocation from certain sources that contain location information. In [16], the authors mined the WHOIS [4] database for IP geolocations. The major issue of using the WHOIS database is that the location information may be outdated or even incorrect, resulting low accuracy as we will see in Section 7.2.

In GeoCluster [18], the authors used the IP location information collected by a large Web portal. The location information was inputted by end users when they were asked to provide their location information by certain location-aware services (such as user registration). The accuracy of GeoCluster therefore depends on the correctness of user input. Another difference between GeoCluster and Structon is that Structon uses publicly available Web pages instead of proprietary data sources.

Structon is also a data-mining-based approach. As compared with previous data-mining-based approaches, Structon takes a new approach by getting IP geolocation from the Web. Structon provides high accuracy and coverage by introducing inference algorithms and integrating IP traceroute.

2.2 Delay-based

There are many schemes that first measure delays to landmarks, then calculate the geolocation (or virtual coordinates) of an IP address based on the measured delays from the end-host to the landmarks [8, 13, 14, 15, 18, 22, 23, 25]. GeoPing [18] maps a host to one of its landmarks based on the measured delays between the landmarks and the host. CBG [8] improves GeoPing by using the measured delays as constrains. The locations derived from CBG need not to be the locations of the landmarks. TBG [13] further improves the delay-based approach by taking advantage of the topology information. In Octant [23], not only positive, but also negative measurement constrains are considered. Since the geographical distance and network delay are only moderately correlated (due to detour routing and queueing and transmission delays), delay-based approaches generally result in hundreds or even thousands of kilometers error distance. There are also approaches that calculate network distance between end-hosts based on certain end-hosts coordinates [3, 17]. The basic operation used by these approaches is also to measure delays between end-hosts.

Structon can be complimentary to the delay-based approaches. The locations of the Web servers determined by Structon can be used by the delay-based approaches as passive landmarks. Since Structon can easily discover the locations of a huge number of Web server IP addresses, the number of landmarks used in delay measurement can be increased in many magnitudes. This will increase the accuracy of the delay-based approaches significantly, since “the error of the class of delay-based algorithms to be strongly determined by the distance to the nearest landmark” [13].

2.3 Grassroot-based

Grassroot-based approach is the third we know for IP geolocation. With this approach, one may setup a community, such as a Web 2.0 site or a BBS forum, and let users to contribute their IP geolocations.

As more and more users get involved, the IP geolocation database will be more complete. Grassroot-based approaches can be very accurate. For example, ip.cn [11], a grassroot site that collects and distributes IP geolocation information, as far as we know, is more accurate than the automated approaches.

This approach, however, has its own issues. First, this approach needs human involvement and the coverage of its database depends highly on the involvement of the community. The ip.cn approach, for example, is a community effort that lasted for more than 6 years. Also it needs constant users’ involvement to evolve. As far as we know, we still do not see a community that can attract the people of the whole world. Hence it seems not practical to build a global IP geolocation database purely based on this grassroot approach. Second, this approach does not work when there are malicious input data. As compared with grassroot-based approaches, Structon is an automatic approach that does not need human input. As we will show in Section 7.2, the accuracy of Structon is close to ip.cn.

There are many commercial companies that provide IP geolocation databases, such as [10, 19]. The technologies they use are commercial secrets, hence it is difficult to compare them with Structon technically. But as we will demonstrate in Section 7.2, Structon provides higher accuracy as compared with an existing commercial IP geolocation database.
3. STRUCTON OVERVIEW

We give the structure of Structon in Fig. 1. It is composed of three parts: Geolocation extraction and clustering, Multi-stage inference, and IP traceroute. The first part mines Web pages from a Web data archive to map Web server IP addresses to geolocations. The second part takes the output of the first part as input and introduces inferring heuristics to get geolocations for more IP addresses. The third part uses IP traceroute measurement to identify the geolocations of IP segments by associating the segments with their access routers. These three parts together address the following two challenges in Structon:

- **Accuracy**: Accurate IP to geolocation mapping given a set of Web pages that may contain wrong geolocation information. We need to: filter false geolocation information contained in Web pages; resolve conflicts when the information in different Web pages of a same host are different; resolve conflicts when different hosts in a same IP address tell different geolocations; and correct errors that are introduced in the Web mining phase. To filter and correct errors, we need to understand and answer where the errors come from. For example, what kind of information in Web pages is more likely to tell the right geolocation of Web servers; what type of Web servers is more likely to tell the correct geolocation of their IP address; what kind of inferring rules can we use to increase the accuracy of our IP geolocation database, given a set of Web server IP addresses and their perhaps wrong locations.

- **Coverage**: Increasing the coverage of the IP to geolocation mapping has two meanings. First, given the geolocations of a set of Web server IP addresses which is only a very small fraction (less than 0.2 percent in this paper) of the whole IP address space, how to cover a significant part of the whole IP address space. Second, given the geolocations of a small set of Web server IP addresses, how to cover a significant part of client IP addresses. Client IP coverage is important, because for many network applications, people care the geolocations of client hosts the most and client hosts are the dominant part of the whole Internet users. We therefore need to demonstrate that Structon covers geolocations for not only Web server IP addresses, but also a significant portion of client IP addresses.

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. The core component of this part is a location extraction algorithm for accurate location extraction. We also perform false information filtering in this part to improve the location extraction accuracy. Since the number of Web pages is huge, we need the extraction algorithm to be of low-time complexity, so that the whole extraction procedure can be finished in a reasonable time.

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 the first stage, we cluster the Web server IP addresses in the same /24 segments to get the location of the /24 segments. We further improve the coverage by studying the location patterns of the segments and by taking advantage of majority voting in the second stage. Finally, we leverage Autonomous System (AS) description information from WHOIS [5] database and Border Gateway Protocol (BGP) [20] routing table to perform further inferring in the third stage.

In the third part, for those IP segments we do not know their locations, we use IP traceroute to find paths from a source to these destination segments. If the locations of the access routers are known in the previous stages, we can deduce the locations of these segments since access router and the corresponding segment are generally located together in the same city.

These three parts then form a complete solution for IP address geolocation mapping. Next, we present these three parts in detail.

Figure 1: The structure of Structon. Structon is composed of three parts. In the first part, we extract geolocations from Web pages and cluster them together to get geolocations for Web server IP addresses. In the second part, we carry out multi-stage geolocation inference heuristics to increase coverage and accuracy ratio. In the third part, we further introduce IP traceroute to increase coverage.
4. 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.

4.1 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 geolocation databases such as zip-codes.com [24].

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. The regular expression library we use is GRETA [7], which is a C++ regular expression library that is very fast for short strings pattern matching. GRETA is very suitable for our purpose, since the size of chunks is typically very short.

Web pages with different charset have different formats to express address, telephone number, and zipcode. For example, in Chinese tradition, city name is always behind province name, whereas in the US, province name follows city name. The zipcode in China is 6 digits, whereas it is 5 digits (or 5 plus 4 digits) in the US. This kind of different can be handled in our geolocation algorithm once we know the conventions for different languages. In what follows, we focus on geolocation extraction for Web pages in China, due to the fact that our Web data were collected from China.

We have built a geolocation table that contains the province and city names, zipcodes and telephone area codes of China. We have 31 provinces (not including Taiwan, Hongkong, and Macao), 508 cities, 486 zipcodes, and 340 telephone area numbers. The pattern matching for zipcodes, city and province names are straightforward. The extracting of telephone area code needs more work, since there are many ways to represent a telephone number. E.g., “8610-22332233”, “8610 22332233”, and “(86-10) 22332233” represent the same telephone number. Our solution is to use multiple patterns for telephone number recognition. In our implementation, we use 10 regular expressions to describe the telephone number patterns. After extraction, the zipcodes and area numbers are all transformed to city names. In case when a zipcode (or area number) is mapped to multiple cities, the first city is chosen. Since in this case, these cities typically are physically co-located, the error distance introduced is quite small.

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). We note that weight assignment is independent of geolocation extraction. 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.

In this part, we filter Web pages that are less trustable to improve extraction accuracy. We filter Web pages that contain specific keywords (“blog”, “bbs”, “forum”) in their urls. This is because the geolocation items contained in these pages are unlikely the geolocations of the Web sites. We also filter the pages that contain a large number of geolocation items (10 in this paper). The reason is that when a page contains many address (zipcode, area code) items, it is very likely to be a Yellow page. The location information contained in Yellow pages is for dissemination or for advertising, and generally cannot be trusted for our purpose.

We also take advantage of the addressing format to further improve extraction accuracy. In Chinese tradition, an address starts from province, then city, then street and building numbers. We take advantage of this to filter false information. For example, in “Addr: Jiangsu province, Nanjing, Tibet road, No. 15”, there are three locations: “Jiangsu” which is a province in China, “Nanjing” which is a city in Jiangsu, and “Tibet” which is a road name (but it is also a province name in China). Since the location of “Tibet” appears behind “Nanjing” and “Jiangsu”, we can safely filter “Tibet” in this case. (Similar rule can be applied to Western style Web pages, though the meaning of position needs to be re-interpreted.)

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.

4.2 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 1.

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 geolocation. For example, in Table 1, the mean weight for Loc_a is therefore (0.64+0.64)/3 = 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.

### 5. MULTI-STAGE INFERENCE

In this paper, we assume that the IP addresses in the same /24 segments are in the same city. A /24 segment contains only 256 IP addresses, hence it is very unlikely that these IP addresses are allocated to two different cities. Besides that, the Internet community also has the tradition to use /24 as the smallest segment allocation granularity. (Even if these IP addresses are allocated to two small cities, these small cities are likely to be close to each other. Hence the geographical error distance should be small in this case.)

In multi-stage inference, the input is a set of Web server IP addresses and their location weight vectors produced in Section 4. 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 a majority of the IP addresses in a /24 say that they are in the same location, we infer that the whole /24 is in that location; 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. We have designed a multi-stage inferring procedure that includes a series of inferring heuristics based on this principle.

Our multi-stage inferring has three stages and these stages are executed sequentially. In Stage I, we perform location calculation for the /24 segments. The purpose of Stage I is to improve accuracy. In Stage II, we perform inferring and error correction based on the location distribution patterns. Stage II increases both coverage and accuracy. We further carry out inferring by leveraging the AS (Autonomous System) description information from the WHOIS database and BGP (Border Gateway Protocol) routing table in Stage III. The purpose of Stage III is to increase the IP geolocation coverage. In what follows, we explain the heuristics and the reasoning behind these heuristics in detail.

#### 5.1 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 2. 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 2, 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_a has the largest probability, we therefore conclude that this segment is in Loc_a.

As we will see in Section 7.2, 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.

<table>
<thead>
<tr>
<th>url \ location</th>
<th>Loc_a</th>
<th>Loc_b</th>
<th>Loc_c</th>
<th>Loc_d</th>
<th>Loc_e</th>
</tr>
</thead>
<tbody>
<tr>
<td>dns_a/url1</td>
<td>0.64</td>
<td>-</td>
<td>0.96</td>
<td>0.89</td>
<td></td>
</tr>
<tr>
<td>dns_a/url2</td>
<td>0.64</td>
<td>-</td>
<td>0.95</td>
<td>0.89</td>
<td></td>
</tr>
<tr>
<td>dns_a/url3</td>
<td>-</td>
<td>0.57</td>
<td>0.95</td>
<td>0.86</td>
<td></td>
</tr>
<tr>
<td>LWV of dns_a</td>
<td>0.43</td>
<td>0.19</td>
<td>0.95</td>
<td>0.88</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Calculating the location weight vector (LWV) of a DNS name. The urls in the table share the same DNS name.

<table>
<thead>
<tr>
<th>IP \ location</th>
<th>Loc_a</th>
<th>Loc_b</th>
<th>Loc_c</th>
<th>Loc_d</th>
<th>Loc_e</th>
</tr>
</thead>
<tbody>
<tr>
<td>61.155.111.42</td>
<td>0.003</td>
<td>0.034</td>
<td>0.003</td>
<td>0.034</td>
<td></td>
</tr>
<tr>
<td>61.155.111.44</td>
<td>-</td>
<td>0.02</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>61.155.111.70</td>
<td>-</td>
<td>0.77</td>
<td>-</td>
<td>0.13</td>
<td></td>
</tr>
<tr>
<td>Location PDF</td>
<td>0.26%</td>
<td>68%</td>
<td>0.26%</td>
<td>20.5%</td>
<td>11%</td>
</tr>
</tbody>
</table>

Table 2: Calculate the location probability distribution function (PDF) of a /24 segment from the location weight vectors of the IP addresses in that segment.
5.2 Stage II: Iterative inferring and error correction

ISPs tend to assign large IP segments (much larger than /24) to a same location. Hence if the locations of the subsegments display a certain pattern, such as the majority of the subsegments are in a same location, it is then very likely that the whole segment is in that majority location. 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. 2. 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 input. The iteration doesn’t end until no further inferring can be performed in both heuristics.

5.2.1 Inferring based on location patterns

The location patterns we use are illustrated in Fig. 2. 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 \( L \);

The reason for case 1 is simple: we have a majority of subsegments (i.e., 75 percent) in \( L \), hence we infer that the whole segment is in \( L \). The reason for case 2 needs more explanation: Since we know the location of only 50 percent of the /24 subsegments, we cannot use majority voting. When the two /24 subsegments are in two different /23 segments, it is a good indication that the whole /22 segment is in \( L \). Since if the other two /24 segments are in different locations other than \( L \), the ISP should assign a whole /23 segment to \( L \) and the rest /23 segment to other locations. This can simply the address allocation and reduce the size of the routing table in routers.

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.

5.2.2 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. \( IPSeg_a \) 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 \( N_a / N_m \geq mthresh_{22} \);
Stage II improves both coverage and accuracy. As we will see later in Section 7.2, Stage II improves the accuracy by more than 10 percent at both province and city levels. And the coverage is improved dramatically from 8.9 million IP addresses to 24.4 million IP addresses using the 500-million Web page data archive.

5.3 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. These two heuristics leverage AS description information in the WHOIS database and BGP routing table. In the first heuristic, we infer locations for small ASes based on their description information in the WHOIS database. In the second heuristic, we leverage the BGP routing table for inferring. BGP routing table contains all the routable IP segments. Our idea here is to use the IP segments in BGP routing table as input and recursively divide them into sub-segments, then use the coverage ratio and majority ratio to decide the location of the sub-segments.

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 [21] provide BGP routing table snapshots. In what follows, we describe these two heuristics in detail.

5.3.1 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 $mthresh_{32}$;
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 $mthresh_{31} = 0.7$. In practise, we found that parameters setting is not difficult, due to the fact that the heuristic is not sensitive to these parameters. See Section 7.2.3 for evaluation of the heuristic under different parameter settings.

5.3.2 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 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. 3.

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. Again, see Section 7.2.3 for evaluation of the heuristic under different parameter settings.

6. TRACEROUTE

As we will show later in Section 7, after the Web mining and inference, there still exist a large number of IP segments that we cannot identify their locations.
/* 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 \( s_{thresh} \)):
        return;
    Calculate \( cratio \) and \( mratio \) for \( S \);
    if (\( cratio \geq c_{thresh} \)) and (\( mratio \geq m_{thresh} \)):
        map \( S \) to \( L \);
        else:
            split \( S \) to two equal sub-segments \( S_1 \) and \( S_2 \);
            RecursiveInferring(\( S_1 \));
            RecursiveInferring(\( S_2 \));
return;

Figure 3: Recursive Inferring using IP segments in BGP routing table as input.

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 routing 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.

Traceroute depends on ICMP echo/response messages. Many ISPs and end hosts turn off ICMP for security reasons. Hence we may not be able to get a complete path for a source/destination pair. For example, if an end host turns ICMP off, we will not be able to tell exactly if the access router appears in the path or not. Our solution to this problem is simple: we only use paths that are complete. In our experiment we found that we can find paths for about 21 percent segments. 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 7. The traceroute and inference can be applied iteratively until the resulted IP geolocation database converges.

7. EXPERIMENTS AND EVALUATIONS

In this section, we first present the experiment result we get by using 500 million Web pages collected from China. Our result shows that by using these Web pages, we finally can get locations for more than 100 million IP addresses. We then evaluate our result in detail. The evaluation includes accuracy and coverage studies, parameter settings for the heuristics, and location error studies.

7.1 Experiments

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 [1]. The Web data set we use contains 276 million static pages and 227 million dynamic ones (we simply say a page is dynamic if its url contains ‘?’). There are 29.4 percent static and 22.5 percent dynamic pages containing location information, respectively. Hence there is no significant difference between these two types of pages for our purpose. The size of the Web data set is approximate 10TB. The dataset is stored in a distributed storage system like GFS [6]. We use Dryad [12], 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 3 shows statistics of the Web data set and the final result we get. The Web data set contains more than 500 millions Web pages and 4 millions unique DNS names. There are more than 124 millions Web pages (and about 550 thousands DNS names) containing geolocation information. The total number of Web server IP addresses is 231,501. The number of Web server
IP addresses that we can get their locations is 157,407. The statistical data shows that one server IP address hosts more than 17 DNS names in average. Hence Web hosting is a common practise. Our final IP geolocation database contains 101M IP addresses, which is 87.3 percent IP addresses assigned to China in March 2008.

7.2 Evaluation

The evaluation contains four parts. We first study the accuracy of our result; we then study the coverage of our IP geolocation database; after that, we study the parameter settings for the heuristics we introduced in Stages II and III; finally, we study how location errors are introduced in Structon.

7.2.1 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 grassroots generated IP geolocation database as our 'groundtruth'. ip.cn contains 120M IP 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). See Section 2 for details.

Using ip.cn as a groundtruth certainly appears to be arguable, because though our experience shows that ip.cn is of high accuracy, we do not know its exact accuracy number. But this is feasible as we demonstrate as follows. We define accuracy ratio of a dataset as the number of IP addresses that are mapped to correct IP addresses divided by the total number of IP addresses in that dataset. Suppose the accuracy ratios of two datasets $D_1$ and $D_2$ are $p_1$ and $p_2$. We also assume that when an IP address is mistakenly mapped to a wrong location, the location is randomly distributed to $M$ locations. Hence for a specific IP address, the probability that the locations in $D_1$ and $D_2$ are the same is $p_1p_2 + (1 - p_1)(1 - p_2)/M$. We call this probability as coherent ratio. Hence if the coherent ratio is high, both $p_1$ and $p_2$ are of high value. Also if coherent ratio between a test dataset and a highly accurate dataset is low, then the accuracy of the test dataset, will be low.

We compare the accuracy of Structon at province and city levels with three geolocation databases: WHOIS, iputil1, iputil2. 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 [16]. iputil1 and iputil2 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 4. 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 [10], 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 5.

We can see that, at the ‘web mining’ stage, the coherent ratios at both province and city levels are not high (73 percent and 66 percent). This demonstrates

<table>
<thead>
<tr>
<th>Number of Web pages</th>
<th>502,880,364</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS names</td>
<td>3,991,164</td>
</tr>
<tr>
<td>Web server IP addr</td>
<td>231,501</td>
</tr>
<tr>
<td>Web pages with location information</td>
<td>124,143,191</td>
</tr>
<tr>
<td>DNS names with location information</td>
<td>549,437</td>
</tr>
<tr>
<td>Mined IP addr with location information</td>
<td>157,407</td>
</tr>
<tr>
<td>Deduced IP addr with location information</td>
<td>100,937,472</td>
</tr>
</tbody>
</table>

Table 3: The results we get from the 500M Web data archive.

<table>
<thead>
<tr>
<th>Database</th>
<th>total_ip</th>
<th>coherent_ratio/overlap_ip</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHOIS</td>
<td>93.8M</td>
<td>0.844/84.6M 0.488/62.6M</td>
</tr>
<tr>
<td>iputil1</td>
<td>72.5M</td>
<td>0.759/65.8M 0.501/56.1M</td>
</tr>
<tr>
<td>iputil2</td>
<td>125M</td>
<td>0.874/98.6M 0.684/72.5M</td>
</tr>
<tr>
<td>Structon</td>
<td>101M</td>
<td>0.935/89.0M 0.874/37.1M</td>
</tr>
</tbody>
</table>

Table 4: 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.
that the correlation of the geolocations of contact information in Web pages and Web server IP is not perfect. But the majority of the Web servers do tell their correct locations. This is important for our Structon project, or majority voting will not be able to hold.

Table 5 also shows that /24 clustering and inference increase coverage as well as accuracy significantly. By using /24 clustering, we can improve the accuracy ratios by more than 10 percent at both city and province levels and improve the coverage by 60 times. The inference heuristics further improve the accuracy ratios by 10 percent and increase the coverage by 5 times.

The final stage ‘traceroute’ can improve only coverage but not accuracy. This is because the accuracy of the locations of the access routers is provided by the inference stage. Hence the ‘traceroute’ stage will not be able to further improve accuracy. Nonetheless, the coverage is improved by more than 2 times.

### 7.2.2 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.)

**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.

Our result therefore clearly demonstrates that Structon covers almost all the client IP addresses. We noticed that the client IP coverage ratio is much higher than 87.3 percent. The reason might be because there are many assigned IP segments never get in real use, whereas the segments identified by Structon are always in active use (due to the fact that we always use active Web server IP addresses in our inferring).

#### 7.2.3 Impact of heuristics parameters

The heuristics we used in the multi-stage inferring introduce several parameters. Traditionally, parameter tuning for heuristics can be tricky. We show that, however, parameter setting in Structon is easy and the heuristics we use are not sensitive to the parameters. We believe that the insensitivity to parameters is a benefit of our data-centric approach: The results become stable and insensitivity to parameter tuning as the data set becomes large.

The heuristic we use in Stage I and the inferring based on location pattern heuristic in Stage II do not need any parameter. Hence we only discuss the parameter settings of the rest heuristics in Stage II and III.

The error correction heuristic in Section 5.2.2 has two parameters, \( nthresh_{22} \) and \( mthresh_{22} \) which defines the minimal number of /24 segments in IPSegList and \( mthresh_{22} \) which defines the minimal value of \( mratio \), which is the ratio for the majority location. We have studied the coherent ratios of the heuristic under different values of \( nthresh_{22} \) and \( mthresh_{22} \). The result is given in Table 6.

Table 6 shows a very interesting phenomenon: larger \( nthresh_{22} \) and \( mthresh_{22} \) actually reduce coherent ratio and infer less IP segments. This is because when we set \( nthresh_{22} \) and \( mthresh_{22} \) to large values, we will reduce the number of IPSegLists and hence reduce the inferred IP segments and the error correction chances. In this paper, we choose \( nthresh_{22} = 10 \) and \( mthresh_{22} = 0.6 \).

The inferring for small ASes heuristic in Section 5.3.1 has two parameters \( nthresh_{31} \) and \( mthresh_{31} \). \( nthresh_{31} \)
We first study the effect of Web hosting. For Web hosting, one Web server IP address may host many Web servers, each with their own DNS names. Web hosting is very common. In our data set, there are 66,486 IP addresses each hosts more than 2 DNS names and 1782 IP addresses each hosts more than 100 DNS names. The DNS name distribution is highly skewed, the first 122661 (or 78%) IP addresses host less than 15% DNS names. Fig. 4 shows the location accuracy provided by Web server IP addresses vs. the number of hosted Web servers. We can see that as the number of hosted Web servers in one IP address increases, the location accuracy provided by the Web servers decreases at both province and city levels.

We next study the location accuracy vs. the number of location items in the location weight vector of a Web server IP address. The result is shown in Fig. 5. The

<table>
<thead>
<tr>
<th>nthresh &lt;sub&gt;31&lt;/sub&gt; (city:prov)</th>
<th>mthresh &lt;sub&gt;31&lt;/sub&gt;</th>
<th>IP addr (10&lt;sup&gt;6&lt;/sup&gt;)</th>
<th>c_ratio (city)</th>
<th>c_ratio (prov)</th>
</tr>
</thead>
<tbody>
<tr>
<td>216:220</td>
<td>0.6</td>
<td>30.1</td>
<td>0.903</td>
<td>0.943</td>
</tr>
<tr>
<td>216:220</td>
<td>0.7</td>
<td>29.8</td>
<td>0.906</td>
<td>0.945</td>
</tr>
<tr>
<td>216:220</td>
<td>0.8</td>
<td>29.4</td>
<td>0.916</td>
<td>0.951</td>
</tr>
<tr>
<td>216:220</td>
<td>0.9</td>
<td>28.0</td>
<td>0.921</td>
<td>0.956</td>
</tr>
<tr>
<td>216:221</td>
<td>0.6</td>
<td>31.9</td>
<td>0.905</td>
<td>0.943</td>
</tr>
<tr>
<td>216:221</td>
<td>0.7</td>
<td>31.6</td>
<td>0.908</td>
<td>0.945</td>
</tr>
<tr>
<td>216:221</td>
<td>0.8</td>
<td>31.3</td>
<td>0.918</td>
<td>0.950</td>
</tr>
<tr>
<td>216:221</td>
<td>0.9</td>
<td>29.9</td>
<td>0.921</td>
<td>0.954</td>
</tr>
</tbody>
</table>

Table 7: Performance of the inferring for small ASes heuristic with different parameter settings. The output of the error correction heuristic in Section 5.2.2 with nthresh<sub>32</sub> = 10 and mthresh<sub>32</sub> = 0.6 is used as the input.

<table>
<thead>
<tr>
<th>cthresh &lt;sub&gt;32&lt;/sub&gt;</th>
<th>mthresh &lt;sub&gt;32&lt;/sub&gt;</th>
<th>IP addr (10&lt;sup&gt;6&lt;/sup&gt;)</th>
<th>c_ratio (city)</th>
<th>c_ratio (prov)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.6</td>
<td>50.0</td>
<td>0.859</td>
<td>0.935</td>
</tr>
<tr>
<td>0.1</td>
<td>0.7</td>
<td>49.4</td>
<td>0.870</td>
<td>0.941</td>
</tr>
<tr>
<td>0.1</td>
<td>0.8</td>
<td>48.7</td>
<td>0.886</td>
<td>0.944</td>
</tr>
<tr>
<td>0.2</td>
<td>0.9</td>
<td>47.0</td>
<td>0.903</td>
<td>0.947</td>
</tr>
<tr>
<td>0.2</td>
<td>0.6</td>
<td>45.6</td>
<td>0.865</td>
<td>0.935</td>
</tr>
<tr>
<td>0.2</td>
<td>0.7</td>
<td>45.2</td>
<td>0.874</td>
<td>0.942</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8</td>
<td>44.8</td>
<td>0.888</td>
<td>0.944</td>
</tr>
<tr>
<td>0.2</td>
<td>0.9</td>
<td>43.5</td>
<td>0.905</td>
<td>0.947</td>
</tr>
</tbody>
</table>

Table 8: Performance of the inferring by leveraging BGP routing table in Section 5.3.2 with different parameter settings. The output of the heuristic in Section 5.3.1 with nthresh<sub>31</sub> = 2<sup>16</sup>:221 and mthresh<sub>31</sub> = 0.7 are used as the input.

![Figure 4: Location accuracy of Web server IPs vs. the number of hosted Web servers.](image)

7.2.4 Analyzing the location errors

Since Structon is a data mining approach, errors are inevitable. In this subsection, we study where the errors come from when we perform geolocation extraction and clustering.
result clearly shows that as the number of location items in a location weight vector increases, the accuracy ratios at both city and province levels all decrease.

Combining results from Fig. 4 and 5, we can see that even for IP addresses that host only one Web server and that are mapped to only one location, the location accuracy is less than 80% at the city level. This clearly demonstrates that we cannot purely depend on the first part of Structon for accurate IP geolocation mapping. Fig. 4 and 5 also suggest that filtering based on the number of DNS name and the size of LWV can improve the location accuracy. We, however, did not use this type of filtering in Structon for two reasons: First, this will reduce Web server IP address coverage significantly since Web hosting and large size LWV are quite common. Second, we find that it is more efficient to improve location accuracy using our inferring heuristics.

We also have studied if the famous Web sites tell the correct geolocations. We got the top 100 famous sites from ChinaRank [2], from which we found 85 sites that are located in China. These 85 sites map to 3685 IP addresses. Our study showed that the geolocation accuracies in these Web sites are 51.4 percent at the city level and 54.5 percent at the province level. The accuracy is significantly lower than that of the average case (the first row of Table 5). The reason is that famous sites tend to distribute their Web servers into different geolocations. When one site is located in several different locations, Structon can tell at most one correct location.

We also have studied the error distance distribution when Web servers tell us wrong locations. The purpose is to study the physical proximity of the real and wrong locations. The result is shown in Fig. 6. The result shows that, when a Web server IP is mapped to a wrong location, the error distance between the real and wrong locations are likely small. For example, the error distances for 50% wrong mapped IP addresses are within 800km, which demonstrated that people tend to choose places near them to host their Web servers.

8. 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. Structon achieves high accuracy by introducing a serial inference and error correction heuristic algorithms. 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.

To the best of our knowledge, Structon is the first that takes a Web mining approach for IP address geolocation. Structon provides high accuracy as well as high coverage due to the rich information provided by the Web and the efficiency of its inference and traceroute methods.

The current implementation of Structon uses Web data in China and the result is promising. A natural issue is how we apply the method developed in Structon for the whole Internet. Besides the Chinese Web sites, we have manually checked many English-language Web sites, and found that many organizations do put their contact information in their Web sites. Based on our limited experiment, we believe that the geolocations in these contact information is also statistically related to the location of the Web server IP address. However,
applying Structon to the whole Internet faces the following issues that we would like to address in our future work:

- There are many languages used in the Internet, hence geolocation extraction for the whole Internet is a very challenge issue that may need collaborations of researchers from different countries.

- More research is needed to find a generic inference granularity for all. In our current implementation, we use city as our inference granularity. This works well for China since China has only several hundreds of cities. But for counties like the US, the number of cities is near 30,000 (and the number of zipcodes is more than 40,000) [24], even though the two countries have similar size. It might be difficult to get enough IP segments to perform efficient inference in this case. Specific for the US, telephone area code seems a more appropriate granularity for location inference, since its number is only 533. But this certainly demonstrates the difficulty in identifying appropriate inference granularity for different countries.

9. 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.

10. REFERENCES