Leveraging Language to Locate Routers: Using LLMs for Parsing rDNS Hostnames

Danish Athar
LUMS
Lahore, PK

ABSTRACT

Geolocating an IP router has many utilities for network operators. State of the art approaches to router geolocation use geohints embedded in router reverse DNS (rDNS) hostnames. But these approaches are purely syntactic and rely heavily on adherence to a consistent router naming convention, which is impossible to guarantee. We use large language models (LLMs) to solve this problem with a natural language processing approach. We conduct experiments using a large pool of real router rDNS hostnames belonging to hundreds of different domains. Since LLMs don't depend on a particular naming convention, and have broader geographic knowledge, our approach not only works where the existing approaches do, but also works in many cases where they fail.

1 INTRODUCTION

Knowing the geographical location of an internet protocol (IP) router has long attracted the interest of networking researchers and service providers. The reasons behind this interest include optimizing network paths for packet delivery [11], providing resilience against natural disasters [13], and avoiding inadvertent censorship [8], among others [9]. Publicly available *IP address to geolocation* databases for end-hosts exist (e.g., MaxMind Geolite [12]), but no parallel exists for internet routers. Creating and updating such a similar list for geolocating the IP routers would require manual labor, making it cumbersome and inaccurate.

Network operators often embed geographic location information in router reverse DNS (rDNS) hostnames to provide location hints (aka *geohints*). This embedded information has previously been exploited by researchers to extract such geohints [9, 16]. Luckie et al. [9] did it by looking at all the available rDNS hostnames in a single domain and crafting a regular expression,[1] which is then used subsequently to extract the geohint from *any* rDNS hostname in that domain. A more recent system, The Aleph[16], uses large language models (LLMs) to group hostname patterns, construct regexes and then map the results of the regex-based extraction to actual geographic locations.

However, such regex-based approaches to extract geohints from rDNS hostnames are prone to inaccuracies owing to two common failure modes when applied at scale. First, these schemes are sensitive to strict adherence to the naming convention, which is impossible to guarantee due to the manual nature of device naming. Secondly, a large number of hostnames from each domain are needed to sufficiently support 'learning' the naming convention used by the operator. These limitations also cause other string-processing approaches (such as rDNS hostname tokenization [14]) to either declare a substring as a geohint inaccurately or miss it altogether.

[1]In our observation, methods like Hoiho and DRoP try to compile a single regex that can be run for all rDNS hostnames in a single domain. A few domains had been assigned two regexes for this same purpose.

Recognizing that the rDNS hostnames for IP routers are manually entered by the network engineers, we argue that IP router geolocation problem is better solved as a natural language problem and posit that large language models (LLMs) can act as universal context-aware parsers that robustly handle naming convention variations within and across service providers. LLMs are also known to work well as zero-shot and few-shot learners, and can thus extract geohints without requiring training and retraining. To demonstrate the feasibility of this approach, we introduce **L^{3}R**, a system that leverages LLMs to translate unstructured hostnames into structured information.

Our approach employs the power of LLMs to significantly simplify the geohint extraction process. Given a list of all the hostnames belonging to a single domain, we instruct the LLM to parse each hostname and try to extract substrings that would classify as geohints. We demonstrate that this methodology allows for the successful extraction of geohints missed by previous approaches, with the potential to also extract a wider array of metadata, including device roles, infrastructure dependencies, and even subtle risk indicators. The system's structured, machine-readable output enables a new level of automated network analysis.

When evaluated over nearly 35000 rDNS hotnames obtained using ITDK toolkit from CAIDA [7], our **L^{3}R** system was successfully able to extract 5462 geohints[2], compared to the 3179 extracted by the state-of-the-art Hoiho framework [9]. We manually checked the hints extracted by Gemini that were not included in Hoiho's accepted geohints for the same domains and found 2185 out of the 2265 additional hints to be acceptable. This significant improvement in geohint extraction comes from the semantic understanding capabilities of LLMs, compared to a regex-based approach that performs pattern matching as a pure syntactic tool, thus making it inherently brittle and highly sensitive to even minor variations in the hostname nomenclature.

We use prompt-engineering techniques like providing few-shot examples and XML tags to get promising results. We also prompt the LLM to provide explanation for each result. Interestingly, the LLM extracted some geohints from which discerning the location information was not obvious to a human observer. However, the *explanation* field clearly identified the corresponding location, as listing 1 indicates.

This paper makes the following contributions:

(1) We frame the challenge of parsing rDNS hostnames as a language-based semantic inference task, better suited to the capabilities of modern LLMs than to traditional, rigid parsing methods.

[2]The geohint extraction is also dependent upon the specific LLM. Over the ITDK data set, Gemini 2.5 Flash extracted more geohints (5462) compared to o4-mini (3840).

Listing 1: Examples of LLM-generated explanations

```
"vl.280.dwari-rt0.mos.com.np": {
  "geohint": "dwarika",
  "explanation": "'dwarika' likely refers to Dwarika's Hotel area, a
  well-known landmark and locality in Kathmandu.",
  "granularity": "locality"
}
"mlx1.stlsmozc-a.socket.net": {
  "geohint": "stlsmozc-a",
  "explanation": "Coded reference to St. Louis, Missouri (STL + MO),
  with additional technical suffix.",
  "granularity": "city"
}
```

(2) We present a preliminary evaluation of **L^{3}R**, a proof-of-concept system that validates this approach. Through a comparative analysis using more than 35000 hostnames from 302 domains, we demonstrate the LLMs' capacity for nuanced analysis, including their ability to utilise all possible information embeddings in the hostnames to extract explainable geohints.

(3) We present a novel, interactive multi-agent workflow tool built on **L^{3}R**'s core principles, enabling complex, on-demand, context-aware analysis of entire network paths, to show further potential of our approach. The code for **L^{3}R** is available openly on GitHub [1].

2 BACKGROUND AND RELATED WORK

Internet Protocol (IP) address geolocation is the process of finding the geographic location of a device, given its IP address or DNS hostname. The device could be an end-host or an infrastructure device such as a router, or a middlebox. IP geolocation is useful for several reasons. For example, a service provider may select a path that does not pass through a specific country for geo-political reasons.

Network operators routinely embed rich (and often human-readable) information about the routers within their reverse DNS (rDNS) hostnames. For example, a hostname like 'if-ae-49-2.tcore2.pvu-paris.as6453.net' signals a location in Paris. These geolocation hints, or 'geohints' within the hostnames can assist the ISP in identifying the physical locations of the corresponding devices[3].

Unfortunately, there is a severe lack of standardization surrounding this information. Operators adopt heterogeneous naming conventions, often embedding varying kinds of informational cues in differing segments of the hostname.

Notable prior approaches to IP geolocation using geohints include DRoP [7], Hoiho, and HLOC [14]. All three approaches begin by constructing extensive databases of *geocodes*—geolocation strings obtained by aggregating location information from various sources[4]. DRoP and Hoiho focus their search on specific segments of hostnames, operating under the assumption that within each autonomous naming domain, geographic hints are applied consistently. This limitation exhibits itself in the failure cases observed by the authors. HLOC lifts this restriction, scanning across the full hostname by comparing all parts to the precompiled dictionary of geocodes, but remains constrained by the completeness of said dictionary, overlooking non-standard location identifiers. Moreover, DRoP and Hoiho split hostnames based on punctuation, causing them to miss geolocation hints that are in the same segment of the hostname without any apparent separator.

Luckie et al. [9] proposed a multi-phase approach for IP geolocation. First, using a pool of hostnames from a domain, they identify a regex to extract potential geohints from a hostname belonging to that domain. Next, they apply that regex to all hostnames in that domain, and match each result against a large pool of location codes to obtain a potential geolocation. Finally, the location is accepted if it satisfies the latency criteria measured from a bunch of nearby probes to the corresponding host. This technique is extremely sensitive to minor variations in naming conventions when assigning rDNS hostnames to routers and middleboxes, thus missing a large number of geohints.

Most recently, Thiagarajan et al. [16] improved on the coverage of tools like Hoiho using large language models. They use LLMs to classify hostnames, generate regular expressions for these classes, and establish hint-to-location mapping per operator. However, we find that the regex usage is still troublesome and acts as a bottleneck in terms of utilising the LLMs' true potential, limiting geohint extraction to certain expected formats by definition.

Our direct-inference LLM-based approach overcomes all these limitations. The language model can interpret hostnames both structurally and semantically, enabling it to detect geolocation cues regardless of position, formatting, or inclusion in a predefined geocode list.

3 METHODOLOGY

3.1 Dataset

Our experiments utilize the publicly available data supplement [10] accompanying Luckie et al.'s work [9], which contains thousands of hostnames from hundreds of network domains, derived from CAIDA's ITDK dataset [4]. This comprehensive dataset provides a diverse collection of real-world router hostnames[5], enabling a direct performance comparison of previous work with our proposed approach.

Our dataset contains 35000 hostnames belonging to 302 distinct domains. Luckie et al.'s approach learns a regex for each of the domains, and then applies it to all the hostnames belonging to the same domain. This extracts possible geohints based on the organization's apparent naming convention. We call the set of these values for all hostnames (belonging to any domain) *extracted potential geohints*. These geohints are filtered through `sc_Hoiho`'s empirical validation process which includes RTT validation, which aims to filter out false positives and ambiguous matches. This results in a more reliable but much smaller set of geohints compared to the raw extracted set. We call these the *accepted geohints*. These are included in Luckie et al.'s dataset as legitimate geolocation indicators.

[3]For ease of management, network operators may also embed information about their devices' roles, hardware, and positions in the network topology within the hostnames.
[4]Sources for location information include airport codes, city and state abbreviations, UN codes for trade and transportation locations, and other naming conventions.

[5]We only considered the entries in the `*-midar-iff.routers` file that included both IP addresses and their corresponding rDNS hostnames, excluding entries containing only IP addresses without hostname information.

3.2 Experiment Setup

We ran one inference per domain, feeding the LLM with all the hostnames for the particular domain to it generating comprehensive domain-wise geohints list for comparitive analysis[6]. We chose two state-of-the-art language models and got two additional data subsets: `gemini25-geohints` (using Google's Gemini 2.5 Flash) and `o4mini-geohints` (using OpenAI's o4-mini) because of their efficient reasoning capabilities and state-of-the-art performance. Both models feature 'dynamic thinking', enabled by default, which lets the model decide the thinking budget to allocate for each input task. On average, we saw around 10000 tokens being used for input, thinking and output per domain. The pipeline is shown in Figure 1.

3.2.1 Prompt Design. As recommended by LLM providers, we used a high-level system prompt [6] that constrained the model's expertise to the scope of our geolocalization problem (shown in Figure 2).

The actual task-level prompt, shown in Figure 3, also follows the best practices for prompt-engineering, such as using XML tags [2] for structure and few-shot examples [3] to guide model behaviour. As a whole, the prompt guides the models through systematic geohint extraction. The prompt defines valid geohints as city and region codes (2-5 letters), ISO country codes, airport codes, UN/LOCODEs, CLLI codes, and recognized location names. For ambiguous cases where regex-based geohint extraction fails, we instruct the models to perform contextual inference by analyzing patterns across the domain's complete hostname collection.

The prompt instructs the model to only return structured JSON output containing the domain name, hostname mappings, extracted geohints, explanations for each identification, and granularity classifications. This standardized format ensures consistent data collection across both model implementations and facilitates automated analysis of results.

4 ANALYSIS

Performance evaluation of our approach is difficult because the ground truth is non-existent. For analysis, we compared our results against the state of the art. We collected the geohints extracted by the LLM against the geohints extracted by running Hoiho's bestregex against the hostnames; and geohints extracted and validated by Luckie et al. (provided in the geohints list for each domain in the data supplement). We also compared our approach's performance with The Aleph [16], which is the latest research development in this space to our knowledge. A detailed text file containing a per-domain comparison of all geohints extracted by each approach is available in our GitHub repository. A few sample comparisons are shown in Tables 1 and 2.

We conducted a direct character to character comparison between the LLM suggested geolocation against the *accepted geohints* from Luckie et al.'s dataset, as well as against The Aleph's extracted geohints for each of the hostnames. Our system's reported performance measures are conservative for two reasons. First, there are cases where the LLM correctly identifies a location but formats it differently than the accepted geohint (e.g., "nyc, new york" vs. "nycny"). By excluding such syntactically different but semantically matching extractions, our system's performance estimate is conservative. Secondly, we believe that in many cases our approach outperforms previous approaches, extracting valid geohints that are discarded or missed by other approaches. Due to such cases, the reported metrics take a hit as the gold standard data we use is based on the previous state-of-the-art (Luckie et al.'s work), and fails to match the performance of our method in these cases.

[6]To keep the length of input prompts, and the returned results, well within the model's context length, we used domains with less than 1000 unique hostnames. Extending this to other domains can simply be done by sending multiple requests to the LLM for different sets of hostnames of the same domain and combining the resultant JSONs.

4.1 Comparison with The Aleph

The Aleph represents an intermediary approach between traditional regex-based extraction and our direct LLM-based method. It leverages large language models to automatically generate regular expressions for hostname parsing, but this means it still relies on deterministic pattern matching for extraction. As a result, it inherits the structural brittleness of regex-based techniques (detailed below in Section 4.2), just simplifying the regex-construction step.

We evaluated The Aleph on the same dataset and domains as our approach. Despite employing LLMs during regex generation, its reliance on precompiled pattern structures leads to substantial information loss. In domains where hostnames exhibit high lexical or positional variability, Aleph frequently fails to identify valid geohints that fall outside its proposed regex's coverage.

Empirically, our model outperforms Aleph across nearly all evaluated domains. Our approach consistently extracted more, and valid, geohints for almost all of about 300 domains. For instance, in the first domain of the dataset (sorted alphabetically by default), *0rbitel.net*, our approach using Gemini identified thirteen valid geohints compared to Aleph's three, with ten unique to Gemini.

Conversely, when Aleph extracts additional strings, these are often false positives. In *aarnet.edu.au*, Aleph produced `aarnet` as a geohint, which is an organization name rather than a geographic location. Similarly, in *accesscomm.ca*, Aleph added tokens such as `core`, `fujitsu`, and `saskenergy`, none of which have geographic meaning. These findings indicate that while Aleph broadens the surface scope of regex coverage, it lacks the semantic filtering and interpretive ability that direct LLM reasoning provides.

Overall, The Aleph is a good case for showing the limits of improving regex pipelines with generative components. While it represents progress toward automation, its extraction process remains constrained by the syntactic rigidity of regex-based geolocation.

4.2 Cases Where LLMs Surpass Regex-Based Extraction

By comparing the different geohint lists for each domain, we were able to manually perform a comparative analysis of our approach versus regex-based pattern matching. Our approach clearly outperformed this traditional method of geohint extraction from hostnames in multiple scenarios, which are detailed below.

4.2.1 Implicit Geolocation Information. Large language models can extract implicit geographic information that traditional regex-based methods fail to capture. These models pack different levels of domain-specific knowledge that they can use to interpret specialized geographic codes, including those that may not be present in standard geographic databases. This gives our LLM-based approach a clear advantage in many instances.

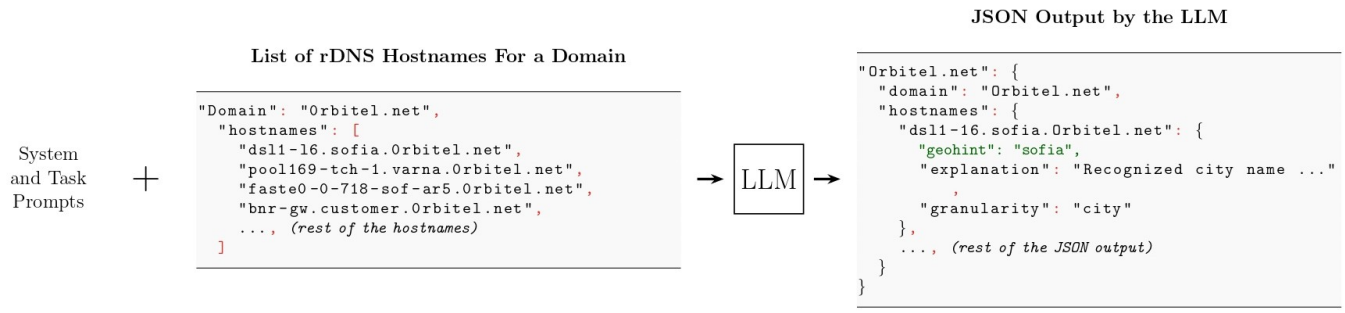

Figure 1: Overview of our proposed LLM-based geohint extraction pipeline. We give a list of all the hostnames for a given domain with our prompts to the LLM and obtain structured JSON output. Only the 'geohint' fields from the LLM's response are used in comparative analysis.

```
"""You are an expert network analyst specializing in reverse DNS hostname interpretation and geolocation analysis.
Your expertise includes:
- Understanding network infrastructure naming conventions
- Recognizing geographic codes in various formats (IATA, ISO, CLLI, UN/LOCODE) and unique naming conventions
- Distinguishing between genuine geographic hints and technical/service identifiers
- Analyzing patterns across multiple hostnames within a domain
Always return valid, structured JSON output that can be directly parsed for further analysis."""
```

Figure 2: System-level instructions provided to the LLMs for geohint extraction.

```
"""For the given domain, analyze each hostname and extract
geolocation hints (geohints) embedded within it.
<Geohint Definition>
Valid geohints include:
- City/region codes (2-5 letters, e.g., "lax", "fra")
- Country codes (ISO 2-letter, e.g., "us", "de")
- Airport codes (3-letter, e.g., "JFK", "CDG")
- UN/LOCODEs (5 characters, e.g., "USNYC")
- CLLI codes (several characters)
- Recognized city/region names (e.g., "tokyo", "sofia")
- Possible shortened/coded names for physical locations
- For unrecognized codes or words, you may also try to infer the
possible location it represents by looking
at the other hostnames for the same domain, but only accept the
geohint if it makes sense
</Geohint Definition>
<Domain Name>
{domain_name}
</Domain Name>
<Hostname data>
{hostname_data}
</Hostname data>
Return structured JSON in the following format:
<Output Structure>
{"domain": "example.com",
"hostnames": {
"nyc1-edge.example.com": {
"geohint": "nyc",
"explanation": "IATA code for New York City",
"granularity": "city"
},
"lon-gw01.example.com": {
"geohint": "gw",
"explanation": "GW likely refers to Gatwick Airport in London since
Lon is present in the hostname which could mean London",
"granularity": "airport"
},
"router-edge-klmzmi01-usa.example.com": {
"geohint": "klmzmi",
"explanation": "CLLI code for Kalamazoo, Michigan. Makes sense
given the USA mention as well.",
"granularity": "area/region"}}}
</Output Structure>
Use the JSON format from the example. Do NOT add markdown
formatting or extra characters."""
```

Figure 3: Prompt used to instruct our LLM-based geohint extraction.

For example, in an Amazon.com hostname [8], a AWS region identifier "us-east-1" was not extracted as a geohint by Hoiho's regex patterns, which appear to focus primarily on IATA airport codes for the hostnames included in the dataset for the Amazon domain. On the contrary, both Gemini 2.5 Flash and o4-mini successfully identified "us-east-1" as a valid geohint, explaining it as 'the AWS region code for US East (Northern Virginia)'.

4.2.2 Non-Standard Geohints. Traditional geohint extraction methods miss non-standard abbreviations and local references that may appear in hostnames. For example, for the domain cybermesa.com, Luckie's method successfully extracted standard CLLI codes like "snfenm" (Santa Fe, NM), but it failed to recognize "simms" as a geographic reference [9] in the *ms1.simms.cybermesa.com* hostname.

[8] *hulksmash.hulksmash.autest.awseb-e-3-....us-east-1.elb.amazon.com*

[9] The 'plan' identified by `sc_Hoiho` for cybermesa states 'CLLI', possibly indicating that their regex for this domain only targets CLLI codes

Domain	Geohints Only in Gemini	Geohints Only in Aleph	In Both
borwood.net	cpt, dur, jhb, lon, lx, mel	-	-
attens.net	ahi, ams, atl, bhx, bki, bos, cdg, chi, dal, fra, hnk, lax, lhr, mia, nyc, phx, pnj, rwc, san, sea, sjc, sng, syd, wdc	-	-
vianet.ca	alliston, bar, blindriver, br, cambrian, elliot lake, goodwood, halib, hunt, kap, kawagama, marshhill, nip, nor, northbay, saultips, sdbry, sud, sudbury, tbay, thames, tor, uxbridge	-	-
space.net	bergkirchen, garching, langenfeld, muc, muenchen, schwaig	-	-
towerstream.com	bos, chi, dal, lax, mia, ny, nyc, ord, sea, sf, sfo	-	-
cnr.it	agrate, bassini, bicocca, fantoli, ge, mariobianco, mi, rm, sangiuliano, segrate, ss	cnr	na
bright.net	arthur, ayers, ayersville, basc, bascom, bryn, celina, champaign, col, crid, fng, glan, indy, lima, maumee, mccl, middlepoint, middlept, minford, newbav, okol, pioneer, shel, spring, stmary, syca, sycamore, toledo, vanw, wabash, wapa, worth	ridge-dhcp, waba-pppoe-pool2	mpnt, ridge, waba
broadbandsolutions.com.au	bne, mel, per, syd	-	-

Table 1: Comparison of geohints identified by Gemini and Aleph across domains shown for 8 random domains from the dataset.

Domain	Geohints Only in Gemini	Geohints Only in Hoiho-Validated [7]	In Both
borwood.net	lx	-	cpt, dur, jhb, lon, mel
attens.net	ahi, bhx, bki, bos, cdg, fra, hnk, lhr, mia, nyc, pnj, rwc, san, sjc, sng, syd	-	ams, atl, chi, dal, lax, phx, sea, wdc
vianet.ca	alliston, bar, blindriver, br, cambrian, elliot lake, goodwood, halib, hunt, kap, kawagama, marshhill, nip, nor, northbay, saultips, sdbry, sud, tbay, thames, tor, uxbridge	frank, pgw, power, shaw	sudbury
space.net	garching, muc, muenchen	-	bergkirchen, langenfeld, schwaig
towerstream.com	chi, ny, ord, sf	-	bos, dal, lax, mia, nyc, sea, sfo
cnr.it	bassini, fantoli, ge, mariobianco, mi, na, rm, ss	-	agrate, bicocca, sangiuliano, segrate
bright.net	ayers, ayersville, basc, bryn, champaign, col, crid, fng, glan, indy, mccl, middlept, mpnt, newbav, okol, shel, stmary, syca, toledo, vanw, waba, wapa	cni, tower, tsc	arthur, bascom, celina, lima, maumee, middlepoint, minford, pioneer, ridge, spring, sycamore, wabash, worth
broadbandsolutions.com.au	-	-	bne, mel, per, syd

Table 2: Comparison of geohints identified by Gemini and Hoiho across domains shown for 8 random domains from the dataset.

Gemini 2.5 Flash correctly identified "simms" as a geohint and explained it to be referring to the Simms Building in Albuquerque, New Mexico, showing its ability to recognize local landmarks and non-standard geographic references. This capability comes from the model's broader knowledge base, which includes information about local buildings, landmarks, and geographic features that are not captured in standardized geographic code databases. Note that The Aleph fails to identify any geohints for this domain, as it does for several others.

Similarly, 3 of the previous approaches mentioned in this paper (HLOC, DRoP and Hoiho) try and extract letters for geohints, which works in the general case. However, this may not work for all domains. The LLMs can intelligently decide to include numbers with the geohint for additional granularity.

The wiscnet.net domain provides an excellent example of this with its use of CESA (Cooperative Educational Service Agency) codes within some of its hostnames. Geohints in hostnames containing identifiers like "cesa06" and "cesa09" would not be recognised by previous approaches, as these codes are alphanumeric and unlikely to appear in standard geographic databases. However, both o4-mini and Gemini 2.5 Flash correctly identified these as references to specific educational service regions in Wisconsin,

with explanations about CESA 6 serving northern Wisconsin and CESA 9 serving south-central Wisconsin. This shows the models' ability to apply specialized knowledge about organizational and administrative geographic divisions as well.

The Aleph does not return any geohints for any hostnames belonging to this domain as well.

4.2.3 Distributed Information Across Hostname Segments. Another case where LLMs shine for this use-case is when geographic information is distributed across multiple segments of a hostname. Regex patterns typically focus on specific positions within period-delimited hostnames, but LLMs can combine information from different parts of the hostname to infer geographic locations.

For example, in the altiusbb.net domain, hostnames like *liberty-core-02.in.altiusbb.net* contain both a city name ("liberty") and a state abbreviation ("in") in separate segments. The LLM successfully recognized this pattern and correctly identified Liberty, Indiana as the geographic location. Similarly, hostnames such as *morris.fl.altiusbb.net* and *el-jobean-bhrtr.fl.altiusbb.net* show the model's ability to combine city or community names with state abbreviations to provide accurate geographic identification. The model interpreted "morris" as referring to Morris, Florida, and "el-jobean" as the census-designated place El Jobean, Florida, both correctly associated with the state abbreviation "fl".

4.2.4 Positional Variance in Geohint Placement. Regex-based methods rely on geohints appearing in consistent positions within hostnames for a given domain. However, real-world hostname conventions can vary significantly, placing geographic identifiers in different positions even for the same domain. The acorus.net domain illustrates this challenge clearly.

While Luckie's regex successfully extracted geohints like "nyc," "par," and "lis" that appeared in consistent positions (occurring around the middle part of each hostname), it failed to identify "seattle" when it appeared at the beginning of the hostname *seattle-ix.wes.acorus.net*. The LLM approach successfully identified all geographic references regardless of their position within the hostname, showing greater flexibility in handling varied hostname structures.

4.2.5 Recovering Valid but Unaccepted Regex Matches. In some cases, it was observed that Luckie's regex patterns successfully extracted geographic strings but these were not included in the accepted geohint list for that domain. The celeritybroadband.com domain shows this issue, where the regex extracted thirteen potential geohints including "mtmrrs," "erwin," and "ftpr," but these were not accepted as valid geographic references. Similarly, the socket.net domain suffers from the same issue for several of the geohints embedded in its hostnames. The extracted geolocation codes are not present in the *geocodes.txt* file given in the data supplement, and they fail the criteria that Luckie et al. have specified for accepting unknown location codes [10].

[10]The LLMs extracted 'cltnmops' as a valid geohint for a hostname belonging to the socket.net domain and explained it to be Clayton, Missouri. This is in line with the other hostnames for this domain, all of which follow a similar naming convention for routers scattered across Missouri. However, Hoiho has excluded all of these geohints. This is likely because the string "cltnmops" contains no substring of four consecutive letters from either of the two words in the identified location (e.g., "clay" from "Clayton" or "miss" from "Missouri") and therefore fails Luckie et al.'s contiguous-substring criterion, which requires at least four letters in a row to match one of the location's component words. This same problem is identified across many hostnames of various domains.

The LLM analysis revealed that several of these rejected extractions were actually valid geographic references. For instance, "mtmrrs" was correctly explained by the LLM as a shortened form of Mount Morris, Illinois, a village within the ISP's service area. This suggests that the LLM approach can also provide valuable validation and interpretation of regex-extracted strings that may have been incorrectly discarded by traditional filtering methods. It also shows how our approach bypasses the limitations of a pre-compiled dictionary of location codes.

4.3 General Positives

4.3.1 Explainability of Extracted Geohints. According to the LLM response structure requested in the prompt, each geohint comes with a detailed explanation of its interpretation, making it possible to understand and validate the extraction logic. This is particularly valuable when dealing with ambiguous or non-standard geographic references, but is also helpful if a human is directly looking at the geohint extraction results.

For example, when extracting "nkz" from *nkz76rtl-po6-251.ll-nkz.zsttk.ru*, the LLM provides the explanation "Short code for Novokuznetsk (Kemerovo Oblast, Russia)," which could be difficult to interpret for someone unfamiliar with such localities. Similarly, "tmk" in *gw-sibintur.ll-tmk.zsttk.ru* is explained as "Short code for Tomsk (Tomsk Oblast, Russia), inferred from other hostnames like gw-tomsksoft." This level of explanation significantly assists the verification and debugging of the extraction process, even if done manually.

4.3.2 Cross-language Geohints. The multilingual capabilities of large language models enable them to recognize geographic references in various languages, expanding the scope of extractable information beyond English-only approaches. In the cyberlink.ch domain, the models successfully identified such geohints, explaining "geneve" as the French name for Geneva and "zuerich" as the German spelling of Zurich.

4.4 Limitations

4.4.1 Cost. Running inferences on state-of-the-art models of reasonable sizes can be a costly endeavor. However, we find that the costs are minimal with the right model choice. For our preliminary evaluation, we were able to stay within the free tier API limits of Gemini 2.5 Flash, completing the analysis for free, at the cost of having to span out our queries according to the rate limits. However, using o4-mini for the same inputs with dynamic thinking enabled and structured JSON output with explanations for each geohint in the output only cost us around 15 USD using OpenAI's API, but without having to wait for restricted free tier rate limits like in Gemini's case.

4.4.2 Variation in Behavior across LLMs. In some cases, it was observed that o4-mini would leave out some of the hostnames given as input. This should be fixed by adjusting the prompt to ask the model to always return all of the input hostnames, even if the model fails to find a geohint in it. However, for sake of our comparison, we kept the prompt exactly the same for both the models, and agree that this is a variation in sensitivity to the prompt's wording expected for different large language models.

4.4.3 Hallucinations. By design, LLMs can produce information that is factually incorrect but still present it to the user as the final output. We tried to minimize its effects by prompting the model to provide explanations about its selected geohint candidates and looking for any inconsistencies.

5 USABILITY

5.1 Multi-Agentic Workflow

To demonstrate how these inference capabilities can be operationalized, we developed a working prototype of an interactive, multi-agent analysis workflow using Google's Agent Development Kit. This system moves beyond static analysis by using a series of specialized agents to handle various kinds of user requests. The workflow can accept inputs such as a single IP address or a full traceroute path, and then dynamically invoke agents for running traceroutes, rDNS lookups, core LLM-based parsing and analysis, and verification steps. This prototype can serve as an easy-to-use context-aware network intelligence tool for other researchers.

To automatically validate and verify the information extracted from the hostnames by the LLM, we added special verification agents in this multi-agentic workflow. These agents automatically use google search (called 'grounding' [5]) and available APIs (like the MaxMind GeoLite endpoint from RIPE) to validate the information extracted by the LLM.

6 FUTURE WORK

We identify the following future directions:

6.1 Using Smaller, Locally Deployed Models

We want to repeat our evaluation using a smaller, locally deployable language model which is particularly important for privacy-sensitive settings where users may be unwilling to share potentially confidential hostname data with commercial LLM providers. It would be interesting to see how well this approach extends to such models.

6.2 Integrating External Geohint Dictionaries via Function Calling

Luckie et al. provide a comprehensive `geocodes.txt` file aggregating location strings from various geohint databases. Future iterations of our system could easily integrate this resource via function-calling APIs, which are now supported by most LLM platforms, to allow the model to cross-reference potential geohints against this dictionary. This would take our model closer to the performance of Hoiho's own validation strategy, which filters candidate strings using this dictionary together with RTT measurements.

6.3 Incorporating Topological Context via GNNs

Geohints from adjacent routers may offer valuable contextual signals. A natural extension is to model router-level graphs and apply Graph Neural Networks (GNNs) to propagate geolocation information across neighboring nodes.

6.4 Structural Interpretability

While our current setup prompts the LLM to explain its reasoning for the extracted geohints, a more robust approach would be to explore interpretability methods grounded in the transformer architecture itself, such as those proposed by Templeton et al. [15], which aim to extract and interpret internal representations of geographic semantics directly from the model's latent space.

6.5 Validation of Extracted Geohints

The promising results of our scheme could be validated by contacting network operators responsible for the domains in the dataset and asking them to verify the locations of the routers as suggested by our extracted geohints. RTT measurements could also be used to attempt to triangulate and confirm the geographical locations of the routers.

7 CONCLUSION

Previous approaches for IP geolocation have relied on employing pattern matching techniques on rDNS hostnames to find parts of the hostname that could be embedding geolocation hints. Such rigid searches for geohints end up missing valid geohints in some cases and extract meaningless substrings in others. To this end, we introduce **L^{3}R** that reimagines this problem as one of natural language processing and uses LLMs' inherent abilities to read and understand text as a direct solution to this problem. With reasonably structured prompting, we show that our approach consistently outperforms previous schemes and aids in human understanding of the extracted substrings as well. This paper serves as a first-step in this direction and naturally has lots of future directions that could now be taken. Importantly, we plan on analysing the performance of smaller models for this purpose and gathering validation feedback from network operators whose routers we have tried to geolocate.

REFERENCES

[1] 2025. L3R: Learning Location from rDNS. https://anonymous.4open.science/r/L3R-hotnets25-05E8/. Anonymous submission to HotNets 2025.

[2] Anthropic. 2025. Use XML tags to structure your prompts. Anthropic documentation. https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/use-xml-tags Accessed July 5, 2025.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL] https://arxiv.org/abs/2005.14165

[4] CAIDA. 2021. Macroscopic Internet Topology Data Kit (ITDK). https://www.caida.org/data/internet-topology-data-kit/. Accessed: 2025-07-05.

[5] Google AI. 2025. Grounding with Google Search – Gemini API Documentation. https://ai.google.dev/gemini-api/docs/google-search. Accessed: 2025-06-19.

[6] Google Cloud. 2025. System instructions. Generative AI on Vertex AI documentation. https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/system-instruction-introduction Accessed July 5, 2025.

[7] Bradley Huffaker, Marina Fomenkov, and kc claffy. 2014. DRoP: DNS-based router positioning. *SIGCOMM Comput. Commun. Rev.* 44, 3 (July 2014), 5–13. https://doi.org/10.1145/2656877.2656879

[8] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. 2009. Nation-State Routing: Censorship, Wiretapping, and BGP. arXiv:0903.3218 [cs.NI] https://arxiv.org/abs/0903.3218

[9] Matthew Luckie, Bradley Huffaker, Alexander Marder, Zachary Bischof, Marianne Fletcher, and K Claffy. 2021. Learning to extract geographic information from internet router hostnames. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies* (Virtual Event,

Germany) *(CoNEXT '21)*. Association for Computing Machinery, New York, NY, USA, 440–453. https://doi.org/10.1145/3485983.3494869

[10] Matthew Luckie, Alexander Marder, Zachary Bischof, Marianne Fletcher, Bradley Huffaker, and k. claffy. 2021. Hoiho Router Naming Conventions per Suffix Over Time (Data Supplement). Data supplement to "Learning to Extract Geographic Information from Internet Router Hostnames". https://publicdata.caida.org/datasets/supplement/2021-conext-hoiho/

[11] Ratul Mahajan, Ming Zhang, Lindsey Poole, and Vivek Pai. 2008. Uncovering Performance Differences Among Backbone ISPs with Netdiff. In *5th USENIX Symposium on Networked Systems Design and Implementation (NSDI '08)*. USENIX Association, San Francisco, CA. https://www.usenix.org/conference/nsdi-08/uncovering-performance-differences-among-backbone-isps-netdiff

[12] MaxMind. 2024. GeoLite2 Free Geolocation Data. https://dev.maxmind.com/geoip/geolite2-free-geolocation-data/#understanding-ip-geolocation Accessed: 2025-07-09.

[13] Ramakrishna Padmanabhan, Aaron Schulman, Dave Levin, and Neil Spring. 2019. Residential links under the weather. In *Proceedings of the ACM Special Interest Group on Data Communication* (Beijing, China) *(SIGCOMM '19)*. Association for Computing Machinery, New York, NY, USA, 145–158. https://doi.org/10.1145/3341302.3342084

[14] Quirin Scheitle, Oliver Gasser, Patrick Sattler, and Georg Carle. 2017. HLOC: Hints-Based Geolocation Leveraging Multiple Measurement Frameworks. arXiv:1706.09331 [cs.NI] https://arxiv.org/abs/1706.09331

[15] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. 2024. Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. *Transformer Circuits Thread* (2024). https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html

[16] Kedar Thiagarajan, Esteban Carisimo, and Fabián E. Bustamante. 2025. The Aleph: Decoding Geographic Information from DNS PTR Records Using Large Language Models. *Proc. ACM Netw.* 3, CoNEXT1, Article 7 (March 2025), 20 pages. https://doi.org/10.1145/3709374