IPHints: Inferring and Categorizing Customers of Internet Services

Rahul Tripathi[1], Katherine Izhikevich[2], Sai Chaparala[2], Manan Patel[2], Samvrit Srinath[2], Eric Zhou[1], Ben Du[1], Liz Izhikevich[1]

[1]UC Los Angeles [2]UC San Diego

Abstract

Internet-facing services often expose hints about their customer. Customer hints help Internet measurement tasks, such as identifying who to contact during vulnerability disclosure. Today, inferring the customer from heterogeneous service data remains a largely manual task. We present IPHints, a system that automatically infers customers of Internet services at scale. For each IP, IPHints extracts customer hints and categorizes the inferred customer type(s), enabling large-scale analysis of Internet services by the kinds of customers they identify. IPHints is four orders of magnitude faster than a human and achieves 95.2% overall accuracy. Within 3.9 hours, IPHints infers customers for services on over 1.8M IPv4 addresses, enabling the largest study to date of customers represented in individual IPv4 Internet services. IPHints is open-sourced under the Apache 2.0 license.

1 Introduction

Internet-facing services frequently reveal information about the entity on whose behalf they might be exposed. A service exposed through Starlink, for example, may present a certificate associated with Netflix: the infrastructure points to the access network, while the service data points to a possible customer. This is not proof of ownership or operational control. However, it can reveal something often more useful for Internet measurement: a customer hint—a service-visible clue about the person, organization, institution, or business for whom the service appears to exist.

Customer hints matter because Internet measurements often need to reason about the party implicated by an exposed service, not merely the infrastructure supporting it. In vulnerability disclosure, the appropriate party to contact may be the customer whose service, data, or users are exposed, particularly when hosting providers lack operational control or jurisdiction over the vulnerable service [4, 11, 25, 35]. In performance studies, customer context can help explain unusual connectivity patterns or outages, such as those experienced by remote customers in open seas [13]. Yet the metadata researchers traditionally use for this purpose—WHOIS contacts [17, 29, 30], routed prefixes, Autonomous Systems [36], hosting providers, or hardware vendors [1]—is only a proxy. It works when IP ownership, service hosting, and customer identity align. Cloud platforms, virtual hosting, IP address leasing, shared access networks, and managed appliances increasingly break this alignment [24].

Prior work has shown that service-level customer hints are often present in exposed service data itself. Researchers have manually inspected cloud-storage artifacts [11], TLS certificates in exposed LEO satellite services [13], and packets traversing GEO satellite links [35] to identify parties associated with exposed services. However, such analyses depend on human interpretation: in our benchmark, manually analyzing only 300 Starlink IPs required 21 hours of researcher effort, making this approach impractical for repeated measurements over millions of Internet-facing services.

This paper therefore explores an unexplored question: can service-visible customer hints be recovered automatically and used to categorize the entities they appear to identify, at Internet scale? Hints are not rare: in a manually labeled benchmark spanning several access networks and a broader IPv4 sample, nearly half of IPs expose at least one discernible customer hint. However, automation must contend with two sources of ambiguity. First, customer hints are not standardized protocol fields and are often mixed with names for vendors, access networks, managed platforms, and default interfaces. Second, even when a hint is recovered, its meaning may remain opaque; an abbreviation, vessel name, or local business string often requires outside context before it can be categorized.

We introduce IPHints, a system that extracts and categorizes one or more customer hints from Internet-facing services at scale. IPHints is built around the observation that hints are often locally repetitive even when globally irregular. Offline, it learns reusable hostname heuristics, including rules that recover customer-controlled labels and rules that strip vendor or platform namespaces. At runtime, it applies these heuristics to compact service fingerprints, uses per-IP inference for cases that fall outside learned rules, and returns Unknown when the remaining evidence is too weak or infrastructure-facing. Finally, because a recovered hint may not reveal what kind of entity it names, IPHints retrieves a limited amount of web context to categorize the inferred customer hint. We evaluate this design against ablations that remove rule matching, stripping, and grounding, showing that no single field or parser is sufficient.

IPHints is four orders of magnitude faster than a human baseline, while achieving 94% customer-hint accuracy

```
Example A: single customer hint on a host
IP: 206.214.224.XXX
Reverse DNS:
   starlink.lax.starlinkisp.net
Routing: 206.214.224.0/24 via
SPACEX-STARLINK (AS14593)
WHOIS: SpaceX Services, Inc.
TLS 443/TCP:
  Subject CN: *.1.oca.nflxvideo.net
  Issuer: Google Trust Services
```

Figure 1: Service-visible hints identify Netflix as the customer. The TLS certificate on TCP port 443 contains `nflxvideo.net`, which hints at Netflix as the customer represented by the exposed service.

```
Example B: multiple customer hints on one IP
IP: 64.127.136.XXX
Routing: 64.127.136.0/22 via
CONNECT2FIRST-LLC (AS399331)
DNS:
   firewall.geeksonline.us
   www.geeksonline.us
   thatledsign.com
   autodiscover.thatledsign.com
   parker-enviro.com
   www.parker-enviro.com
TLS 443/TCP:
   Subject CN: www.geeksonline.us
   SANs include: geeksonline.us,
www.parker-enviro.com
   Issuer: Sectigo RSA Domain Validation
Secure Server CA
```

Figure 2: One IPv4 address can expose multiple customer hints—DNS and TLS names hint to several distinct customer-facing domains and therefore customers.

and 95% categorization accuracy against a manually labeled benchmark. It also correctly abstains on 97% of indiscernible cases, avoiding unsupported customer claims from generic infrastructure strings. At scale, IPHints processes 1.8M IPv4 addresses in 3.9 hours, making customer-hint inference practical for large Internet measurements.

Customer hints change how Internet measurements are interpreted. Applying IPHints across access networks reveals that exposed services point to markedly different customer mixes; for example, satellite access infrastructure exposes visible hints associated with maritime, transport, utility, and public-safety services that would be difficult to distinguish from infrastructure metadata alone. This kind of context can help researchers prioritize disclosure, explain performance anomalies, and understand which kinds of parties appear in exposed-service measurements. More broadly, IPHints turns service-visible artifacts into a scalable measurement primitive: one that lets Internet measurement reason not only about where services are hosted, but on whose behalf they might be exposed.

2 Defining a Customer Hint

The data returned by a service often reveals context beyond the application itself. Observable features of an Internet-facing service can provide hints about the entity on whose behalf the service appears to be exposed. We refer to that entity, as represented in the service-visible data, as the *customer* of the service. A *customer hint* is a service-visible clue that points to this customer.

Figure 1 illustrates a public IP address whose service data hints at Netflix as the customer for whom the service is exposed. For IP address 206.214.224.XXX, the service running on TCP port 443 presents a TLS certificate containing `nflxvideo.net`, which hints at Netflix since "nflx" is the stock ticker symbol for the company. The same service data also contains information about other parties involved in exposing that service—the reverse DNS and routed prefix point to Starlink infrastructure—who are not the customer.

A single public IP address may hint at multiple customers. This can occur in shared hosting, NAT, port forwarding, or other multi-tenant settings where multiple organizations or services are represented behind the same address. Figure 2 hints at multiple customers. The service data for IP address 64.127.136.XXX include DNS and TLS references to several distinct customer-facing domains, including `geeksonline.us`, `thatledsign.com`, and `parker-enviro.com`. The TLS certificate subject names `www.geeksonline.us`, the SANs also include `www.parker-enviro.com`, and the DNS records expose names under all three domains. This illustrates a shared or multi-tenant environment in which the address hints at multiple customers.

In other cases, the service data does not hint at the customer at all, and instead hints only at the surrounding infrastructure used to expose the service. Figure 3 contains DNS and routing information pointing to Starlink infrastructure, while the forward DNS, TLS names, HTTP content, and software strings point to Peplink-managed administrative interfaces. In this setting, the customer is *indiscernible*: the service data reveals only supporting infrastructure, not the entity for whom the service is exposed.

3 Customer Hints in the Wild

Three observations motivate IPHints. First, customer hints are prevalent enough to enable large-scale analysis. Second, customer hints appear across multiple sources, including

```
Example C: indiscernible customer
IP: 148.222.196.XXX
Reverse DNS:
   *.pop.starlinkisp.net
Forward DNS:
   1934-791e-ae01.mypep.link
Routing: 148.222.196.0/24 via
SPACEX-STARLINK (AS14593), PE
TLS 443/TCP:
  Subject CN: 1934-791e-ae01.mypep.link
  Additional names: captive-portal.peplink.com,
  www.captive-portal.peplink.com
HTTP 443/TCP: title = Web Administration
Interface
HTTP headers: server = nginx; CSP includes
*.peplink.com
Software: Peplink router, nginx
```

Figure 3: Generic service data can make the customer indiscernible— The exposed data identifies Starlink infrastructure and Peplink-managed interfaces, but not the customer of the exposed service.

DNS, TLS, and application-layer service data, rather than in any single source alone. Third, although researchers can manually interpret customer hints from service data, doing so is simply too slow to scale.

Methodology. We constructed a dataset of 1,500 unique IPv4 addresses drawn from Censys snapshots on May 8, 2025 and November 6, 2025, each with at least one responsive service on the corresponding snapshot date [7].

The dataset was constructed from four groups: 300 random hosts with services from Starlink (AS14593), 450 from Google Fiber (AS16591), 300 from PUNTONET (AS22724), and 450 from a broader Internet-wide sample without AS-level filtering.

Each IP address was independently evaluated by two researchers, who determined whether the available service-exposed data contained a customer hint and, if so, recorded that hint along with the hint's business categories from the NAICSlite taxonomy [36]. The taxonomy provides 13 top-level business categories and 92 more specific second-level categories nested under them, enumerated in Appendix D. A third researcher resolved disagreements. Annotators did not use AI assistance, but were allowed to consult external sources when needed to verify whether a candidate string plausibly referred to the customer of the exposed service rather than to a vendor artifact, supporting infrastructure, or an unrelated token. To minimize potential optimism bias, the architect and evaluator of IPHints did not participate in constructing this ground truth dataset.

Dataset	DNS	TLS	HTTP	WHOIS	SMTP	L2TP	POP3	IMAP	N
PUNTONET	86.4	98.4	98.4	98.4	98.4	100.0	100.0	100.0	191
Starlink	96.9	99.4	100.0	100.0	100.0	100.0	100.0	100.0	163
Google Fiber	86.5	98.2	100.0	100.0	100.0	100.0	100.0	100.0	163
Random IPv4	60.5	96.1	96.6	100.0	100.0	100.0	100.0	100.0	205
Combined	81.4	97.9	98.6	99.6	99.6	100.0	100.0	100.0	722

Table 1: **Cumulative hint coverage**—The table reports, for the 722 discernible IPs of the total 1500 IPs, how often customer hints appear in service-data fields when fields are added in order. DNS alone accounts for 81% of this subset, and adding TLS raises that fraction to 98%.

Results. Across the 1,500 labeled IPs, 722 IPs hint to at least one discernible customer, while 778 are indiscernible. The distribution is uneven across our pooled groups: Google Fiber contains 163 out of 450 (36.2%), the unrestricted Random IPv4 set contains 205 out of 450 (45.6%), Starlink contains 163 discernible-customer IPs out of 300 sampled (54.3%), and PUNTONET contains 191 out of 300 (63.7%). Together, these results show that discernible customers are commonly found in service exposed data. We use both the discernible and indiscernible customer results to evaluate IPHints's accuracy in Section 9.

Manual labeling is possible and reliable, but time-consuming. For the Starlink sample, annotators required an average of 2.1 minutes per IP to determine whether a customer was hinted and, when hinted, to assign a category. Two independent annotations of just 300 Starlink IPs required 21 hours of analysis. The time required is large enough to make manual extraction impractical at scale, even before considering broader Internet coverage or snapshots over time.

The cumulative cover-set analysis in Table 1 shows that customer hints cannot be recovered reliably from any single service-data field alone. DNS is the strongest single field, providing 81.4% of IPs with discernible customer hints in the structured subset (588/722), but it does not provide complete coverage. Adding TLS raises cumulative coverage to 97.9% (707/722). The same pattern holds across datasets, although the relative contribution of DNS and TLS differs.

Our results show that customer hints are a common and structured feature of exposed-service data. Hints appear often enough across multiple network settings to support large-scale analysis, researchers can manually label them, and hints usually appear in a small number of service-data fields. Most IPs with discernible customer hints contain only one or two distinct hints, and DNS together with TLS accounts for nearly all such coverage. This combination of prevalence, manual labeling cost, and concentration motivates the existence of IPHints.

Domains yielding a hint		Domains without a hint	
Registered domain	**Share (%)**	**Registered domain**	**Share (%)**
synology.me	2.43	peplink.com	28.32
quickconnect.to	2.12	mypep.link	2.43
fortiddns.com	1.65	dtx.io	2.12
dynamic-m.com	1.38	draytek.com	1.95
infinityfleet.net	1.09	welcome.online	1.93
ddns.net	0.79	fritz.box	1.48
dyndns.org	0.55	myfritz.box	1.47
accutrac.io	0.39	myfritz.net	1.45

Table 2: **Top grouped domains with and without usable hint rules in Starlink**— We show a subset of the top 25 domains, which in aggregate account for 57% of all observed forward-DNS and TLS hostnames, yet only 8 (left-column) yield a usable extraction rule. The remaining frequent domains (right column) are still important, as identifying domains that do not contain hints allows us to forgo processing for a substantial number of hosts.

Hint location	**%**	**Example**
Leftmost label	18.4%	**kccourier**.ex.com
Registered-domain label	62.0%	vpn.**identity**.com
Elsewhere in hostname	10.7%	f.b.**identity**.ex.com
Prefix before delimiter	1.3%	**identity**-vpn.ex.com
TLS subject org or CN	4.4%	Org=**identity**
Other TLS metadata	0.5%	email=x@**identity**.com
Other protocol sources	2.2%	HTTP banner / SMTP / IMAP

Table 3: **Customer hints appear in multiple places**— Hint evidence appears across hostname positions, TLS metadata, and other protocol fields.

4 Learning Hint Structure

Many Internet-facing services hint customers in DNS or TLS-derived hostnames. In this section, we characterize where customer hints appear in hostnames and how repeated hostname structure supports learned heuristics for hint recovery.

Customer hints often appear in hostnames, but not in a single consistent structural position. We define a hostname to be any domain name exposed through DNS (including forward and reverse names) or TLS service fields (including SAN and certificate fields). Table 3 shows that, across the 722 discernible IPs from Section 3, hint evidence appears at different positions within DNS names, TLS metadata, and other protocol fields. The most common location is the registered-domain label (62.0%), followed by the leftmost label (18.4%), elsewhere in the hostname (10.7%), and prefixes before delimiters (1.3%). Smaller fractions appear in TLS subject fields (4.4%), other TLS metadata (0.5%), or other protocol sources such as HTTP, SMTP, or IMAP (2.2%). No single hostname-parsing rule can therefore recover customer hints reliably.

Hint placement is often specific to the registered domain. For example, names under `fortiddns.com` often place the customer hint in the label immediately before the registered domain, while names under `dynamic-m.com` may encode it as a readable prefix before an opaque suffix. In `captive-portal.peplink.com`, no usable customer hint is present, so a hostname-derived rule will fail. Table 2 shows the top 10 most common registered domains in the Starlink dataset and the share of hostnames they account for. In Starlink, the top 25 registered domains account for 57% of all observed forward-DNS and TLS hostnames, but only 8 of those 25 yield a usable rule. We observe similar concentration in other networks as well, including Google Fiber and T-Mobile. This concentration allows IPHints to learn reusable heuristics for dominant registered domains, including when to focus on domains that expose customer hints and when to filter out those that do not.

Nevertheless, repeated hostname structure does not cover all cases. This remaining set includes both infrequent registered domains and cases where the customer hint appears outside DNS-derived structure altogether. Some hints appear in TLS subject fields, certificate metadata, HTTP banners, SMTP strings, IMAP strings, or other application-layer service data. Repeated hostname structure therefore helps, but it does not cover every source of customer evidence on its own.

Many services hint at the customer in a hostname, but those hints are not expressed in a single uniform way. Some registered domains repeatedly place the customer hint in a predictable position, while others embed it in provider-managed syntax or fail to expose a customer hint at all. IPHints therefore must learn hostname-derived heuristics where repeated domain-specific structure is common, including both rules that recover customer hints and rules that identify non-informative namespaces (Section 7.1). For the much larger set of cases that fall outside reusable hostname rules, and for cases where the customer hint is not in the hostname, IPHints must rely on more flexible inference capabilities (Section 7.2).

5 Categorizing Customer Hints

Recovering a customer hint is not always enough to determine what kind of customer it refers to. Many customer hints are proper nouns, abbreviations, internal identifiers, or otherwise context-dependent names that are difficult to interpret in isolation. A customer hint such as *stapem boreale* does not itself reveal that it refers to a maritime vessel [21]. Likewise, strings such as *LUTUF* or *Deep Vision* do not directly indicate whether the customer is a business, a school, a government office, or something else. In each of these cases, the recovered customer hint identifies a plausible customer without providing enough information to determine anything substantial about the customer, such as a business category.

Categories make customer hints useful for large-scale Internet measurement. While a customer hint can identify the customer of an exposed service, categorization makes it possible to aggregate those hints across services, compare patterns across sectors, and study the Internet through the kinds of customers its services point to.

Reliable categorization therefore requires more than the customer hint alone. Associated web pages, domain-associated content, or other public descriptions will be necessary to contextualize who the customer is. Again, a hint such as *stapem boreale* does not itself reveal any connection to maritime vessels, so categorization requires finding the vessel in Marine Traffic [21]. In this work, that additional information is defined as *external context*. Section 7.3 describes how IPHints enriches recovered customer hints with this context, and Section 9.2 empirically shows why it is necessary.

6 System Requirements

Given an Internet-exposed service, IPHints aims to recover one or more customer hints and categorize the inferred customer. IPHints must satisfy the following requirements:

Fast. IPHints must be faster than a human analyst. This requires minimizing expensive per-IP reasoning and making effective use of repeated structure in service-exposed data.

Cost Efficient. IPHints must keep the financial cost per-IP inference manageable. Naively, applying a powerful model, such as a commercial large language model, to every IP would be prohibitively expensive. Thus, expensive inference must be reserved for cases where it is truly needed, with prompts and outputs kept compact.

Accessible. IPHints must be usable by the Internet measurement community without requiring proprietary benchmark data or custom model training. This requires relying on broadly available inputs and modest compute resources.

Accurate. IPHints must be reliable enough to support downstream Internet measurement analysis. Since model outputs may be unsupported or unstable, the system should ground decisions in evidence, check correctness where possible, and leave uncertain cases unresolved rather than force incorrect answers.

Scalable. IPHints must scale to millions of IPs. This requires reusing work across repeated patterns, batching unresolved cases efficiently, and avoiding repeated computation wherever possible.

7 System Architecture

In this section, we introduce IPHints, a system that infers customer hints from Internet services and categorizes the inferred customer. IPHints uses a staged pipeline, illustrated in Figure 5. IPHints first learns heuristics in an offline stage.

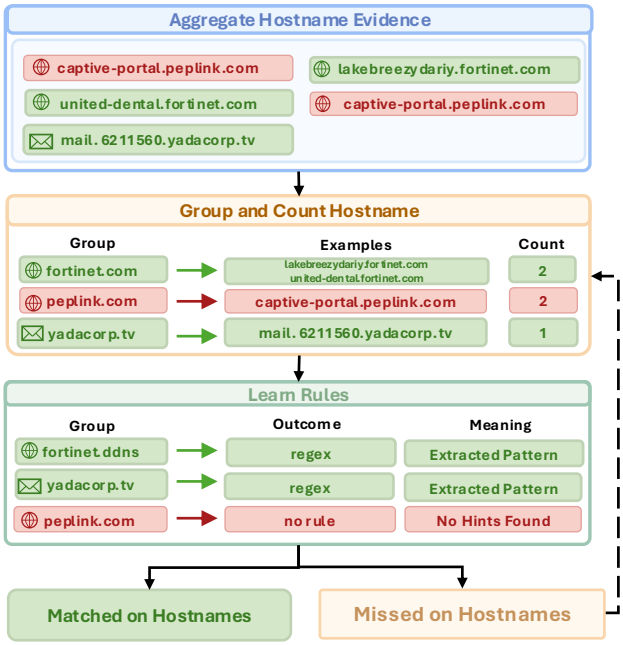

Figure 4: Offline hint-rule learning induces reusable hostname heuristics—IPHints aggregates hostnames across IPs, groups hostnames by registered domain, and ranks common domains before learning whether each domain should recover a hint, be stripped as a false domain, or receive no reusable rule.

It then applies those heuristics to recover customer hints from service-exposed data. Next, IPHints uses the recovered hints to infer the customer. Finally, IPHints categorizes the inferred customer using additional information from the web. Section 8 describes a fast, scalable, and efficient implementation of this architecture.

7.1 Offline Learning Heuristics

Customer hints appear in multiple positions within a hostname (Section 4). However, customer-hint placement often follows repeated domain-specific patterns that can be learned. IPHints therefore learns heuristics over groups of IPs that share the same registered domain, as shown in Figure 4. To learn heuristics, IPHints aggregates the observed hostnames across the target group of IPs and groups them by registered domain. It then ranks registered domains by frequency and focuses on the most prevalent ones. For each prevalent registered domain, IPHints learns a domain-specific heuristic. These heuristics are then reused when processing individual IPs, allowing this stage to be performed offline.

Rahul Tripathi[1], Katherine Izhikevich[2], Sai Chaparala[2], Manan Patel[2], Samvrit Srinath[2], Eric Zhou[1], Ben Du[1], Liz Izhikevich[1]
[1]UC Los Angeles [2]UC San Diego

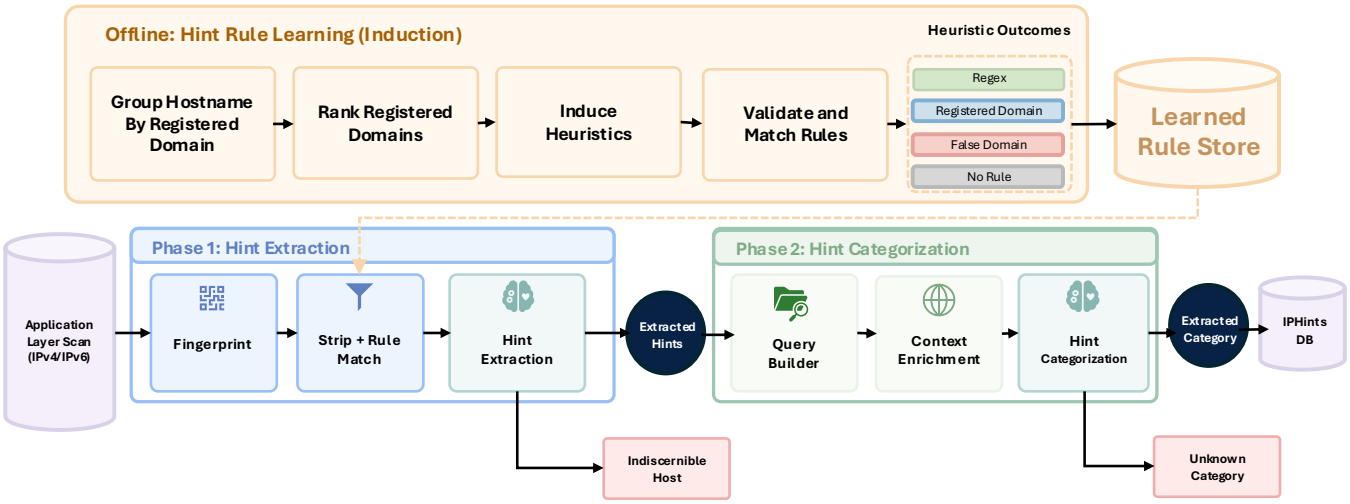

Figure 5: IPHints pipeline—IPHints first learns reusable hint-recovery rules from repeated hostname structure in an offline stage. At runtime, it applies those rules together with per-IP inference to recover customer hints from service-exposed data, then retrieves a limited amount of web information to categorize the inferred customer.

IPHints returns one of four heuristic types: *registered domain*, *regex*, *no rule*, or *false domain*. A *registered domain* heuristic indicates that the registered domain itself is the strongest hint. A *regex* heuristic indicates that one or more extraction rules can recover candidate hints from recurring hostname structure. A *no rule* heuristic indicates that the observed hostnames are too sparse, noisy, or heterogeneous to support a reusable rule. A *false domain* heuristic marks domains that are frequent, but uninformative for customer extraction, such as vendor, CDN, or relay domains.

7.2 Hint Extraction

Once IPHints learns hint-recovery heuristics for prevalent domains, it is ready to recover customer hints from the service-exposed data of individual IPs. IPHints converts the raw service data of each IP into a compact fingerprint that retains the fields found to be most informative in the benchmark from Section 3. If an IP exposes none of these fields, IPHints marks it as `indiscernible`. IPHints then cleans the fingerprint by applying learned heuristics from frequent registered domains to: (1) recover candidate hints from recurring hostname structure ("rule match") and (2) suppress content known not to carry useful customer hints ("strip"). If the fingerprint returns empty after this stage (e.g., a heuristic stripped all the remaining content), IPHints again marks the IP as `indiscernible`.

IPHints then uses the cleaned fingerprint together with any heuristic-derived hints to *extract* one or more customer hints. If multiple customer hints are extracted, IPHints returns them all. The output of this stage is a set of extracted customer hints together with the remaining service-exposed data needed for categorization.

```
Raw fingerprint
IP: 162.240.54.144
TLS: CN=6211560.yadacorp.tv
SMTP: Exim on 6211560.yadacorp.tv
POP3/IMAP: Dovecot ready
DNS: mail.6211560.yadacorp.tv,
captive-portal.peplink.com
```

```
After stripping and rule matching
IP: 162.240.54.144
TLS: CN=6211560.yadacorp.tv
SMTP: Exim on 6211560.yadacorp.tv
POP3/IMAP: Dovecot ready
DNS: mail.6211560.yadacorp.tv
Recovered hint: yadacorp.tv
Illustrative rule:
  ^mail.[^.]+.(?P<hint>[^.]+.[^.]+)\$
Suppressed: captive-portal.peplink.com
```

Figure 6: Stripping removes platform artifacts while preserving customer-hint fields—Repeated fields containing `yadacorp.tv` are retained, while platform-managed names and generic captive-portal text are suppressed.

7.3 Hint Categorization

Next, IPHints categorizes the customer indicated by the recovered hint. As discussed in Section 5, a customer hint alone is often insufficient for reliable categorization. IPHints therefore uses additional information from the web to infer the category of the recovered customer.

7.3.1 Context Enrichment IPHints enriches the recovered customer hint(s) with web information before categorization. It uses the cleaned fingerprint produced during hint extraction along with the customer hint, The goal of this stage is to connect the customer hint to web content that clarifies what kind of customer it refers to. This design follows the broader retrieval-augmented paradigm [12, 14, 16], where non-parametric evidence is used to ground inference when the input itself is too underspecified to support a reliable prediction from parametric knowledge alone.

IPHints then issues deterministic web queries derived from that record to recover the business or organizational context needed for categorization, such as products, services, or institutional role. When a plausible domain name is available in the fingerprint, IPHints uses it to guide retrieval. Section 8 describes how IPHints limits the amount of web information retained for each IP. The output of this stage is an enriched customer hint consisting of the recovered hint together with the retrieved web information needed for categorization.

7.3.2 Categorization IPHints maps each enriched hint produced by the context-enrichment stage to a business category. IPHints first narrows the set of possible categories by ranking candidate labels against the enriched hint. The highest-ranked categories are then used to select a final category, or `Unknown` when the remaining information is insufficient. The output of this stage is a ranked category list from the category taxonomy.

8 Implementation

IPHints is implemented as a staged pipeline that mirrors the architecture in Section 7. The main implementation challenge is making multi-stage customer-hint inference practical at Internet-measurement scale: the system must learn reusable heuristics over groups of IPs, process individual IPs efficiently at runtime, and keep retrieval and model inference efficient enough to run over large IP populations. The implementation therefore combines deterministic preprocessing and retrieval with self-hosted model inference. Censys scan records [7] provide the raw input to the runtime pipeline.

Offline Learning Heuristics. To amortize heuristic induction over repeated hostname structure, heuristic learning is implemented as a batch process over the most frequent registered domains observed in the target group of IPs. Forward-DNS names and TLS SAN/CN names are grouped by registered domain (eTLD+1), ranked by frequency, and the top 500 registered domains are selected for induction. For each selected domain, a compact input is constructed containing up to 20 representative forward-DNS and TLS hostname samples. `gpt-5.4` is used with a fixed 8-shot prompt (Appendix B.1) to assign one of the four heuristic types defined in Section 7.1. We use a commercially hosted model for this offline stage because heuristic induction is run once per target group rather than per IP, making its cost small relative to the full pipeline while allowing the induced rules to be reused across many matching hostnames. The few-shot examples in the prompt teach the model that useful hints may appear as a single label, as part of a deeper delegated name, or as a readable substring embedded within a provider-managed hostname.

Learned heuristics are validated for syntactic correctness and coverage by testing whether they compile and satisfy the provided should-match and should-not-match examples. If the model returns *regex*, the returned expression must expose the recovered hint through the required capture group. When the initial rules explain only part of the observed hostnames, IPHints groups the remaining unmatched names into simple structural buckets and reruns induction on those residual patterns.

Hint Extraction. To reduce per-IP inference cost, hint extraction is implemented as a batched local inference pipeline over compact fingerprints. The service data of each IP is converted into a fingerprint that retains DNS names, TLS certificate fields, and any ASCII application-layer data (HTTP, L2TP, SMTP, and related fields), together with any candidate hints recovered by the learned heuristics. Figure 6 illustrates the fingerprint before and after cleaning. The fingerprint is cleaned by suppressing content associated with registered domains returned as `false_domain`, thereby removing vendor and default strings, and by rewriting shared-namespace hostnames into candidate hints using the learned heuristic bundle. As shown in Figure 6, if the fingerprint contains `site.yadacorp.fortiddns.com`, the learned heuristic for `fortiddns.com` can recover `yadacorp` as a candidate hint. We evaluate the efficacy of cleaning in Section 9.1.

The cleaned fingerprint is then converted into a compact prompt context. A fixed 9-shot prompt and deterministic decoding (`temperature=0`) are used with a local Gemma 4 26B checkpoint served through vLLM [15]. Because the prompt template is shared across queries, vLLM can batch inference across many fingerprints and reuse common prompt

Dataset	Hint TP		Hint FN		Abstain TP		Abstain FP		N
	% (count)	95% CI	% (count)	95% CI	% (count)	95% CI	% (count)	95% CI	
PUNTONET	97.4 (186/191)	[94.0, 98.9]	2.6 (5/191)	[1.1, 6.0]	100.0 (109/109)	[96.6, 100.0]	2.1 (4/191)	[0.8, 5.3]	300
Starlink	93.3 (152/163)	[88.3, 96.2]	6.7 (11/163)	[3.8, 11.7]	89.8 (123/137)	[83.6, 93.8]	4.3 (7/163)	[2.1, 8.6]	300
Google Fiber	92.0 (150/163)	[86.8, 95.3]	8.0 (13/163)	[4.7, 13.2]	98.6 (283/287)	[96.5, 99.5]	6.7 (11/163)	[3.8, 11.7]	450
Random IPv4	94.6 (194/205)	[90.6, 97.0]	5.4 (11/205)	[3.0, 9.4]	97.6 (239/245)	[94.8, 98.9]	3.9 (8/205)	[2.0, 7.5]	450
Combined	**94.5** (682/722)	[92.5, 95.9]	5.5 (40/722)	[4.1, 7.5]	96.9 (754/778)	[95.5, 97.9]	4.2 (30/722)	[2.9, 5.9]	1500

Table 4: **Hint extraction accuracy**— Across 1,500 labeled IPs, IPHints matches at least one labeled customer hint for 94.5% of discernible IPs and correctly abstains on 96.9% of indiscernible IPs. Hint TP uses the any-candidate-match criterion; Hint FN denotes no matching extracted candidate; Abstain FP denotes over-abstention on discernible IPs. Bracketed intervals are 95% Wilson confidence intervals.

prefixes. In practice, this allows the extraction stage to process many IPs efficiently without issuing one hosted-model request per IP.

Context Enrichment. To retrieve enough web information for categorization while keeping per-IP retrieval compact, context enrichment is implemented as a deterministic query-building stage over the recovered hint and the remaining fingerprint content. IPHints generates up to 4 web queries per IP. These include quoted and unquoted hint queries, hint-plus-domain queries, direct domain queries, and site-bounded queries such as `site:domain`. When the remaining domain corresponds to a provider-managed platform, the query builder does not use that domain as the primary query anchor. The resulting queries are issued to the Brave Search API [3]. Only the top 5 normalized results per query are retained, including titles, URLs, snippets, and age when available. IPHints stores the issued queries and retained normalized results for each IP as a JSONL retrieval artifact. This makes the categorization input auditable and allows evaluated queries to be pulled from stored retrieval evidence rather than relying only on live search.l

Categorization. To keep category assignment constrained and consistent across many ambiguous customer hints, IPHints uses the ASdb NAICSlite taxonomy [36] for categorization. Before final classification, IPHints scores candidate NAICSlite categories against the enriched record using a separate cross-encoder reranker based on `cross-encoder/ms-marco-MiniLM-L-6-v2`. This produces a ranked category list for the final prompt. A ranked candidate set is used before final selection because many recovered hints support several nearby categories, and resolving that ambiguity is more reliable when the final step compares a small set of plausible labels rather than committing immediately to one, as shown later in Table 6.

The categorization prompt asks the model to assign exactly one label from the allowed NAICSlite taxonomy, together with a ranked top-3 list, while returning `Unknown` when the remaining information is insufficient. Deterministic decoding (`temperature=0`) is used with the same local Gemma 4 26B checkpoint used in hint extraction. The returned JSON is parsed and normalized before use. Post-processing validates that the predicted label lies in the allowed taxonomy, repairs malformed ranked lists when necessary, and falls back to `Unknown` when the output cannot be normalized reliably.

IPHints is open sourced under the Apache 2.0 license. Customer-hint heuristics are released under `[redacted]`.

9 Evaluation

IPHints identifies customer hints with 94% true-positive accuracy, assigns the correct customer category in 92% of cases, and processes millions of IPs with services in a few hours while keeping per-IP pipeline cost below a tenth of a cent. To minimize potential optimism bias, the researcher evaluating IPHints was not involved in the original labeling of the evaluation dataset (Section 3).

9.1 Hint Extraction

Hint extraction has two distinct failure modes: IPHints can recover the wrong customer hint when one is present, or it can incorrectly decide that no customer hint is present. We therefore evaluate across two dimensions: positive-case accuracy and abstention behavior. Then, we perform an ablation study to show how stripping and learned rule matching affect each part of the pipeline.

Methodology. Hint true positives (Hint TP) measure performance on IPs for which the ground truth contains a customer hint and IPHints extracts at least one matching hint. Hint false negatives (Hint FN) are IPs with a ground-truth customer hint for which no extracted hint from IPHints matches. Abstain true positives (Abstain TP) measure performance on IPs for which the ground truth contains no

Method	Hint Acc.	Abstain Acc.	Overall
Full extractor	94.5 [92.5, 95.9]	96.9 [95.5, 97.9]	95.7 [94.6, 96.6]
No stripping	86.7 [84.0, 89.0]	39.1 [35.7, 42.5]	62.0 [59.5, 64.4]
No rule matching	90.2 [87.8, 92.1]	37.1 [33.8, 40.6]	62.7 [60.2, 65.1]

Table 5: **Stripping and rule matching improve abstention**— The table reports hint-extraction ablations with 95% Wilson confidence intervals on our combined ground truth.

customer hint and IPHints correctly abstains. Abstain false positives (Abstain FP) are for IPs where the ground truth contains a customer hint, but IPHints incorrectly abstains. Overall accuracy combines both discernible and indiscernible cases.

Accuracy Results. IPHints achieves between 92.0% and 97.4% true-positive hint accuracy, depending on the network. Table 4 shows that across all measures of accuracy, abstention accuracy varies most noticeably (89.8%–100.0%). The variation suggests that the harder part of hint extraction is not recovering customer hints when a discernible hint exists, but deciding when hints are too weak or ambiguous to support any customer hint. Across datasets, hint false negatives stay low, while abstain true positives vary more, reinforcing that abstention rather than hint recovery is the main remaining challenge.

Ablation: Rule Matching and Stripping. Table 5 shows that stripping and learned rule matching primarily improve abstention. On the ablation benchmark, removing stripping lowers hint accuracy from 94.5% to 86.7%, lowers abstention accuracy from 96.9% to 39.1%, and lowers overall accuracy from 95.7% to 62.0%. Removing rule matching produces a similar pattern: hint accuracy changes less sharply, falling to 90.2%, while abstention accuracy falls to 37.1% and overall accuracy falls to 62.7%. These results show that the main risk in extraction is treating generic infrastructure or service strings as customer hints when the correct behavior is to abstain. Stripping and rule matching therefore work together: stripping removes strings that do not hint at the customer, while rule matching restores customer hints that can be recovered from repeated hostname structure.

Error Analysis. Most remaining extraction failures occur when the service-exposed data hints only at supporting infrastructure or vendor-managed services rather than at the customer. In a random sample of unresolved cases, the largest share consisted of scans containing only vendor or appliance information (43.3%), followed by generic provider-managed hostnames (30.0%). Smaller fractions exposed no usable hint signal at all (16.7%) or only default or local TLS strings (10.0%). These results suggest that the remaining difficulty is dominated by IPs whose service-exposed data does not reveal the customer in the first place.

Dataset	Top-1	Top-3	2nd-1	2nd-3	N
IPs with Customer Hints Only					
PUNTONET	84.3	95.8	75.4	92.1	191
Starlink	81.0	88.3	66.9	84.0	163
Google Fiber	79.8	95.7	77.3	90.2	163
Random IPv4	86.3	100.0	76.6	94.6	205
Combined	83.1	95.3	74.2	90.6	722
All Ground Truth					
PUNTONET	94.0	98.3	88.0	97.0	300
Starlink	92.3	95.7	87.0	93.0	300
Google Fiber	96.4	100.0	94.0	97.8	450
Random IPv4	96.7	100.0	92.7	98.2	450
Combined	**95.2**	**98.8**	**91.0**	**96.8**	1500

Table 6: **IPHints assigns customer categories accurately**— On the full 1,500-IP ground truth, IPHints assigns the correct top-level category in 95.2% of cases and includes it in the top three predictions in 98.8% of cases. Accuracy is lower on IPs with customer hints only, where `Unknown` abstentions are excluded. Top-1 and Top-3 report accuracy for the 13 top-level NAICSlite categories; 2nd Top-1 and 2nd Top-3 report accuracy for the 92 more specific second-level categories.

9.2 Categorization

We evaluate categorization accuracy and conduct an ablation study on how using external context affects accuracy. We use the common machine learning term *grounding* to refer to the use of external context from the web.

Methodology. We evaluate categorization accuracy at both levels of ASDB's NAICSlite taxonomy: 13 top-level categories (Appendix Table 11) and 92 more specific second-level categories nested under them (Appendix Table 12). At each level, we report both top-1 and top-3 accuracy. Top-1 accuracy counts a prediction as correct only when the highest-ranked category matches the ground truth. Top-3 accuracy counts a prediction as correct when the ground-truth category appears anywhere among the three highest-ranked categories.

The distinction between top-1 and top-3 accuracy matters because customer categorization is inherently ambiguous [6, 36]. A healthcare-related hint may plausibly fit *Hospitals and Clinics* or *Diagnostics and Laboratories*; a maritime hint may fit *Commercial and Workboat*, *Water Transport and*

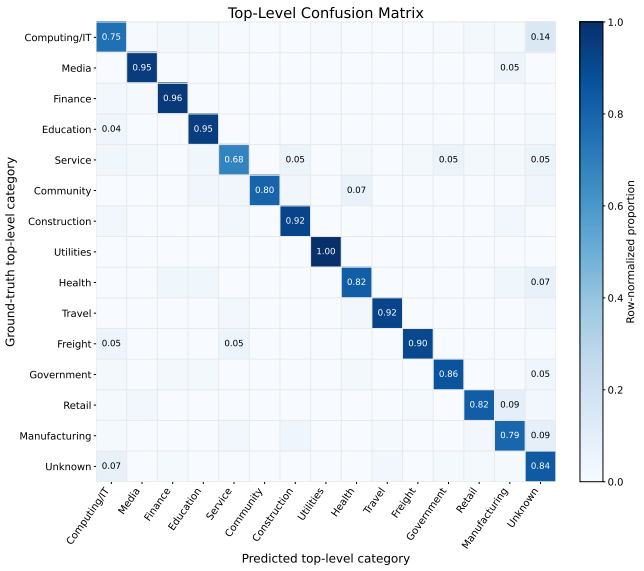

Figure 7: Top-level categorization is largely accurate. Across all labeled datasets, the confusion matrix is dominated by diagonal entries. The main remaining errors are predictions of *Unknown* or confusion between closely related service categories. Cells show row-normalized percentages; only non-trivial cells are annotated.

Shipping, or *Private or Individually Named Vessel*. Top-3 accuracy therefore measures whether IPHints surfaces the correct category among the most plausible candidates, even when its highest-ranked prediction differs from the human label in Section 3. For proportion-based metrics shown with intervals, we report 95% Wilson confidence intervals.

Grounded Categorization Accuracy. Table 6 reports grounded categorization accuracy across the four labeled datasets. Grounded categorization performs well across all four datasets, with the strongest top-level top-1 accuracy on Random IPv4 (96.7%) and PUNTONET (94.0%), followed by Google Fiber (96.4%) and Starlink (92.3%). When IPHints misses the exact ground-truth category, it often returns *Unknown* or confuses related sectors rather than making arbitrary predictions. We illustrate which categories IPHints most commonly confuses in Figure 7. When just filtering for discernible IPs (i.e., IPs that have a customer hint and known categorization), IPHints achieves between 84–90% accuracy top-level top-1 accuracy depending upon the network.

Second-level categorization is harder than top-level categorization overall because IPHints now must choose from 92 possible categories instead of simply 13. IPHints top-1 second-level accuracy ranges from 87% (Starlink)–94% (Google Fiber), with an overall combined accuracy of 91%

Method	Top-1	2nd-1
Hint only: TF-IDF	15.1	26.5
Hint only: Bi-encoder	17.1	22.4
Hint only: Bi-encoder + reranker	44.1	25.9
Hint only: LLM	46.8	37.1
Grounding + reranker	55.1	34.0
Grounding + reranker + LLM	83.1	74.2

Table 7: **Grounded categorization improves accuracy across all datasets.** The table reports top-1 categorization accuracy over the combined benchmark of 722 labeled discernible IPs. Grounding with reranking and LLM selection reaches 83.1% top-level accuracy and 74.2% second-level accuracy across all datasets.

However, when analyzing its top-3 accuracy, IPHints's accuracy improves to a combined accuracy of 96.8%. For second level categorization, we show in Table 8 the most common tuples of top 3 categories when IPHints top 3 categories includes the correct category, which demonstrates that the chosen categories are very similar. Prior work similarly found that the NAICSlite taxonomy can introduce ambiguity between closely related categories [6].

Overall, these results show that external grounding works well for top-level categorization, while second-level categorization remains harder. In Appendix C.1, we evaluate how sensitive categorization is to external context that was retrieved not at runtime.

Ablation: No Grounding. We evaluate the accuracy of categorization without using external context and only using the hint. We compare several hint-only categorization baselines on the labeled set introduced in Section 3. These baselines use only the extracted hint text and do not use any external context. As shown in the first four rows of Table 7, we include lexical matching with TF-IDF, semantic matching with a bi-encoder, retrieval followed by reranking, and direct LLM prediction from the hint alone.

Table 7 shows that categorizing from the hint alone is difficult. Across the combined benchmark, hint-only methods reach only 15.1% to 46.8% top-level top-1 accuracy and 22.4% to 37.1% second-level top-1 accuracy. Reranking helps, and direct LLM prediction from the hint alone does better than the non-LLM hint-only baselines, but the results remain well below the grounded methods. This suggests that the hint often does not contain enough information on its own for accurate categorization. Even with broad parametric knowledge, LLMs struggle when hints are sparse, abbreviated, or weakly represented in public text. Unlike the extraction problem in Section 4, the challenge here is not finding the right hint, but figuring out what that hint refers to in the real world.

Category 1	Category 2	Category 3	% IPs
Buildings and Civil Engineering	Other Manufacturing	–	8.5
Banking and Lending	Investments and Funds	–	6.8
Buildings and Civil Engineering	Machinery and Industrial Production	–	5.9
Connectivity and Telecom	Software, Security, and IT Services	–	5.9
Hosting and Cloud	Software, Security, and IT Services	–	5.9

Table 8: **Top-3 alternatives concentrate in recurring category patterns**— For rows where the correct second-level label appears only in the top three, we group predictions by the unordered set of categories returned in the top-3. The most common sets involve closely related categories, and several contain only two returned labels, indicating that IPHints often narrows the ambiguity to a small set rather than guessing.

Stage / metric	Starlink N = 43,864	Google Fiber N = 64,416	50K Sample N = 50,000	PUNTONET N = 20,972	Free SAS N = 1,822,383
Pipeline yield					
Raw	100.0%	100.0%	100.0%	100.0%	100.0%
Fingerprint	53.0%	44.3%	66.6%	93.5%	94.0%
Offline heuristics	24.7%	29.8%	58.9%	89.1%	15.5%
Model hinting	24.7%	11.2%	48.8%	3.3%	10.5%
Enrichment	8.6%	10.8%	23.9%	3.3%	1.0%
Categorized	8.6%	10.8%	23.9%	3.3%	1.0%
Runtime					
Wall-clock time	8.6 min	11.2 min	9.4 min	5.7 min	3 h 52 min
Time / 1K raw IPs	11.8 s	10.4 s	11.3 s	16.3 s	7.6 s
Cost					
LLM inference	$0.53	$0.69	$0.57	$0.35	$14.19
Search	$18.76	$34.92	$59.67	$3.43	$86.84
Total	$19.29	$35.60	$60.24	$3.77	$101.03
Total / 1K raw IPs	$0.44	$0.55	$1.21	$0.18	$0.06

Table 9: **Pipeline yield, runtime, and cost**—Pipeline-yield rows report the percentage of raw IPs remaining after each stage, with the raw input size shown under each network name. Runtime and cost are reported both as totals and normalized by the number of raw IPs. Search accounts for most monetary cost, while LLM inference remains below $0.02 per 1K raw IPs in every run.

9.3 Scalability

IPHints's scalability is constrained by the later reasoning stages, which are the most expensive because they require per-IP model inference and, for categorization, external context retrieval. Scalability therefore depends first on limiting how many IPs reach those stages, then on how much repeated hostname structure can be exploited before they are reached, and finally on making the remaining reasoning efficient.

IPHints first scales by reducing how many IPs reach the later reasoning stages at all. Table 9 shows that the largest reductions occur before external context is retrieved and categories are assigned. Fingerprinting first removes IPs that do not expose enough service-exposed data to support downstream inference, leaving 53.0% of Starlink, 44.3% of Google Fiber, and 66.6% of the 50K sample. After model hinting, external context retrieval and categorization are applied only to the remaining 8.6% of Starlink, 10.8% of Google Fiber, and 23.9% of the 50K sample. Final categorized outputs are produced for only 8.6% of raw Starlink IPs, 10.8% of raw Google Fiber IPs, and 23.9% of the 50K sample. These reductions show that IPHints achieves scalability first by narrowing the set of IPs that require the most expensive later-stage reasoning.

How much additional reduction IPHints achieves then depends on how much shared hostname structure is available. In Starlink and Google Fiber, the offline heuristics stage reduces the remaining IPs to 24.7% and 29.8%, respectively, by removing non-hint strings and recovering hints from repeated hostname structure. The 50K sample also passes through the same offline-heuristics stage, but because it is not tied to a single network with prelearned hostname patterns, that stage cannot exploit the same kind of AS-specific reusable rules. This difference is important for scalability: when many IPs share dominant registered domains, learned heuristics remove more IPs before later reasoning, directly reducing the time and cost of running the system. When that shared structure is weaker, fewer IPs can be filtered in this stage, and more must proceed to the expensive later stages. Consequently, the 50K sample is the most expensive setting on a per-IP basis, at approximately $0.00121 per IP, which is about 2.2× higher than the next most expensive dataset (Google Fiber at $0.00055 per IP).

Model choice then determines whether those remaining reasoning stages can run at scale. IPHints uses the open-source Gemma models in the later stages because they preserve nearly the same accuracy as the strongest closed-source models while offering much better cost and throughput. In hint extraction, Gemma 4 26B reaches about 88.8% accuracy, compared with about 88.1% for the strongest closed-source model, while costing well under $1 per 1K IPs instead of roughly $15. In categorization, the Gemma models reach about 79% top-level accuracy, compared with about 80% for the strongest closed-source models, but run much faster: Gemma 4 26B processes roughly 3600–3800 tokens/s, whereas the comparable closed-source models are closer to 1200–2100 tokens/s. Appendix C.2 gives the full comparison. Together, these results show that IPHints's scalability depends on both reducing how many IPs reach later reasoning and reducing the cost of reasoning over the IPs that remain.

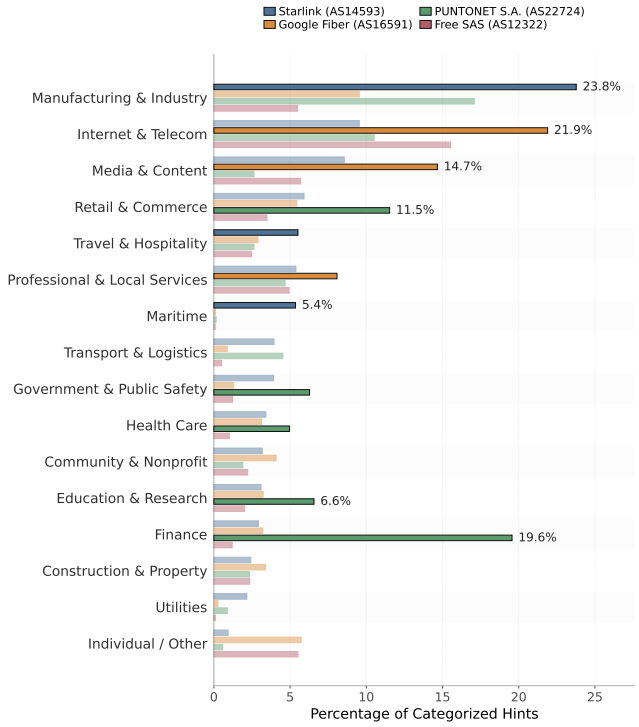

Figure 8: Top-level category distributions across access networks— Customer hints reveal different exposed-service mixes: Starlink has visible maritime, transport, utility, and public-safety hints; among non-*Unknown* categories, Google Fiber and Free SAS are more concentrated in Internet and media categories; and PUNTONET is concentrated in Finance.

IPHints is scalable: it identifies customer hints in 1.8 million IPs with services belonging to the Free SAS ASN 12322 in under 4 hours. Across all networks, on average, IPHints takes 11 seconds to infer customers for 1K IPs. Thus, IPHints is 16,293× faster than the human annotators in Section 3.

10 Using Customer Hints in Measurement

We demonstrate three use cases of IPHints. First, we show how IPHints's business categorization can be used for aggregate network analysis. Then, we show an example of how IPHints can identify potential vulnerable customers or how it can offer explanations for unusual performance.

Aggregate Network Analysis. Figure 8 shows that these visible customer categories differ sharply across networks. Starlink stands out because its categorized hints include many more maritime, transport, utility, and public-safety-associated services than the other access ASes. Starlink maritime customer hints include commercial and workboat names, individually named vessels, water transport and shipping, fishing vessels, and ferry, charter, or tour boats.

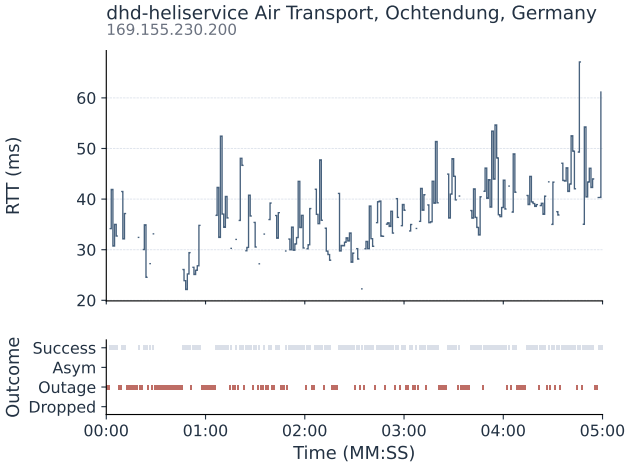

Figure 9: Customer context helps interpret anomalous Starlink performance— For *dhd-heliservice*, categorized by IPHints as *Air Transport*, per-second probes show 37% outages and highly variable RTT over a five-minute window.

Other networks show different customer hint concentrations. Google Fiber and Free SAS are more concentrated in Internet and media categories, with many hints associated with Internet and telecom, media and content, software, and individually operated services. In contrast, PUNTONET is more concentrated in finance, along with finance-adjacent and institutional categories such as banking, insurance, government, and education. Notably, for networks such as Google Fiber and PUNTONET, there is little in their AS-level characteristics that would suggest these customer distributions, making systems like IPHints necessary to uncover them.

Vulnerability Prioritization. Customer hints can help prioritize vulnerability discovery and disclosure. IPHints recovers a customer hint `eseguaviare` from a TLS hint "CN=www.eseguaviare.gov.co" and categorizes it as *Hospitals and Clinics*. The scan of that host shows a public PostgreSQL listener on port 5432, a FortiGate-associated login surface on port 10443, and HTTP on port 8530 consistent with Microsoft HTTPAPI and possibly WSUS. UK government security guidance treats internet-exposed PostgreSQL on port 5432 as a risk because it can permit direct access to the database engine [33]. The United States Cybersecurity and Infrastructure Security Agency has published alerts on Fortinet post-exploitation activity tied to known FortiGate vulnerabilities and on a critical WSUS vulnerability [8, 9]. The fact that the customer hint points to a hospital-associated entity makes the same set of exposures more concerning, as they may provide a path into a hospital network. Knowing the potential customer can therefore help prioritize remediation and route disclosure to the appropriate entity.

Unusual Performance. Categorized customer hints can also help explain observed performance. IPHints discovers a categorized *Air Transport* customer, `dhd-heliservice` from a DNS hint "dhd-heliservice.de," within the Starlink network. We measure the performance of this host using the open-source Roman-HitchHiking tool [32] and plot the measurements in Figure 9. Over the five-minute window, the host records outages 37% of the time and highly variable round-trip time. On its own, the trace is just another exposed host facing persistent outages and highly variable round-trip times. However, knowing that the service belongs to a helicopter business suggests follow-up work to confirm whether the Starlink dish is actually on the helicopter, which may help explain the unusual connectivity, even for LEO networks [13].

11 Related Work

IPHints builds on four areas of prior work: manual interpretation of scan artifacts, automated infrastructure classification, hostname-based inference, and LLM-based IP categorization. While these approaches motivate our design, they differ from our focus on extracting and categorizing service-level hints from Internet-facing systems.

Early work demonstrates that application-layer artifacts can reveal service ownership, but typically relies on manual analysis. *Hitchhiking* [13] identifies Starlink terminals from scan metadata, while Liu et al. [18] use MX records to map email infrastructure to customers. Other studies analyze web artifacts to infer S3 ownership [11] or tracking infrastructure [26]. These approaches highlight the presence of identity signals in scan data, but are difficult to scale due to manual interpretation.

A second line of work uses machine learning to classify infrastructure into coarse organizational or industry categories. Prior systems classify ASes based on routing behavior [10] or map them to organizations using registry and business metadata [2, 5, 36]. Related efforts apply NLP to website content and certificates for attribution [28], and more recent work uses LLMs to label IPs from metadata such as DNS, WHOIS, and TLS fields [23]. Unlike these approaches, which directly assign broad labels, IPHints separates hint extraction from categorization. It first recovers concrete, customer hints from scan data, then maps them into a taxonomy—improving interpretability, auditability, and reuse.

Hostname-structure-based methods show that naming regularities can reveal network metadata [19, 20]. However, these approaches rely on consistent patterns across networks, whereas our setting is less structured: hints may be sparse, embedded in provider-specific formats, or absent from hostnames entirely. Recent work also uses LLMs to extract geographic hints from hostname structures [31]. IPHints differs by targeting customer-facing hints: identifiers that reflect the entity associated with the exposed service, rather than the host's location or network role.

Finally, our categorization stage builds on retrieval- and tool-augmented language models, which improve factuality by grounding outputs in external evidence [12, 14, 16, 22, 27, 34]. This is particularly important in our setting, where hints often involve long-tail entities that are difficult to interpret from strings alone.

12 Limitations

IPHints is limited by what publicly accessible service data can reveal. In many cases, that data clearly points to a particular organization, institution, business, or other party, and in this work we refer to that party as the customer of the service. That customer is not always the direct operator of the service, since a third party may host or manage content on the customer's behalf. Many IPs do not expose any usable customer hint, and others expose only weak, generic, or conflicting evidence. This is especially common in managed platforms, vendor-controlled namespaces, and sparse scan records, where visible DNS, TLS, and application-layer information often reflects supporting infrastructure rather than the customer. Any analysis built on customer hints will therefore be biased toward the customers that are visible in publicly accessible service data.

13 Conclusion

We introduced IPHints, a system that infers customer hints from Internet-facing services and categorizes the inferred customer. IPHints is motivated by a simple gap in Internet measurement: provider- and infrastructure-level information does not always reveal who a service is for. By learning reusable hostname heuristics, recovering customer hints from service-exposed data, and using a limited amount of web information for categorization, IPHints turns measurement data into customer-level measurements. Our results show that customer hints are common enough to support large-scale analysis, that hint extraction requires more than a single global rule, and that accurate categorization depends on more than the customer hint alone. Across our ground truth, IPHints achieves strong extraction and categorization accuracy while remaining efficient enough to run at Internet-measurement scale.

We are actively working on deploying IPHints across the entire IPv4 address space with public facing services (approximately 200 million addresses according to Censys). Because the pipeline separates reusable offline learning from per-IP inference, it can be parallelized across additional compute as needed. More broadly, we hope IPHints helps make customer-level measurements easier to inspect, audit, and

reuse in future Internet measurement research. By releasing IPHints and the accompanying customer-hint annotations, we aim to support new studies of who Internet-facing services are for, how those customers differ across networks, and how that additional context can inform security and performance analysis.

References

[1] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, et al. 2017. Understanding the Mirai botnet. In *USENIX Security Symposium*.

[2] A. Baumann and B. Fabian. 2014. Who Runs the Internet? Classifying Autonomous Systems into Industries. In *Proceedings of the 2014 International Conference on Web Information Systems and Technologies*. SciTePress, 79–90.

[3] Brave. 2026. Brave Search API. https://brave.com/search/api/ Accessed: 2026-04-20.

[4] Jack Cable, Drew Gregory, Liz Izhikevich, and Zakir Durumeric. 2021. Stratosphere: Finding vulnerable cloud storage buckets. In *Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses*. 399–411.

[5] Xue Cai, John Heidemann, Balachander Krishnamurthy, and Walter Willinger. 2010. Towards an AS-to-organization map. In *Proceedings of the ACM Internet Measurement Conference (IMC)*. ACM, 199–205.

[6] Zhiyi Chen, Zachary S. Bischof, Cecilia Testart, and Alberto Dainotti. 2026. AS2Biz: Leveraging Web Presence and AI to Improve AS Business Classification. In *Proceedings of the 2026 ACM Internet Measurement Conference (IMC '26)*. Association for Computing Machinery, Karlsruhe, Germany. To appear.

[7] Hudson Clark, Jeff Cody, Elliot Cubit, Zakir Durumeric, Matt Ellison, Liz Izhikevich, and Ariana Mirian. 2025. Censys: A Map of Internet Hosts and Services. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*.

[8] Cybersecurity and Infrastructure Security Agency. 2025. Fortinet Releases Advisory on New Post-Exploitation Technique for Known Vulnerabilities. https://www.cisa.gov/news-events/alerts/2025/04/11/fortinet-releases-advisory-new-post-exploitation-technique-known-vulnerabilities. Accessed: 2026-04-27.

[9] Cybersecurity and Infrastructure Security Agency. 2025. Microsoft Releases Out-of-Band Security Update to Mitigate Windows Server Update Service Vulnerability, CVE-2025-59287. https://www.cisa.gov/news-events/alerts/2025/10/24/microsoft-releases-out-band-security-update-mitigate-windows-server-update-service-vulnerability-cve. Updated October 29, 2025. Accessed: 2026-04-27.

[10] Xenofontas Dimitropoulos, Dmitri Krioukov, George Riley, and kc claffy. 2006. Revealing the Autonomous System taxonomy: The machine learning approach. In *Passive and Active Network Measurement Workshop (PAM)* (Adelaide, Australia).

[11] Soufian El Yadmani, Olga Gadyatskaya, and Yury Zhauniarovich. 2025. The File That Contained the Keys Has Been Removed: An Empirical Analysis of Secret Leaks in Cloud Buckets and Responsible Disclosure Outcomes. In *Proceedings of the 2025 Conference on Cloud Security*. https://project-theseus.nl/publication/2025/the-file-that-contained/the-file-that-contained.pdf

[12] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.* 24, 1, Article 251 (Jan. 2023), 43 pages.

[13] Liz Izhikevich, Manda Tran, Katherine Izhikevich, Gautam Akiwate, and Zakir Durumeric. 2024. Democratizing LEO Satellite Network Measurement. *Proc. of the 8th ACM POMACS* (2024).

[14] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large Language Models Struggle to Learn Long-Tail Knowledge. arXiv:2211.08411 [cs.CL] https://arxiv.org/abs/2211.08411

[15] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles* (Koblenz, Germany) *(SOSP '23)*. Association for Computing Machinery, New York, NY, USA, 611–626. https://doi.org/10.1145/3600006.3613165

[16] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 793, 16 pages.

[17] Frank Li, Gilbert Wondracek, Marco Balduzzi, Engin Kirda, Christopher Kruegel, and Giovanni Vigna. 2016. You've Got Vulnerability: Exploring Effective Vulnerability Disclosure. In *25th USENIX Security Symposium*. USENIX Association.

[18] Enze Liu, Gautam Akiwate, Mattijs Jonker, Ariana Mirian, Stefan Savage, and Geoffrey M. Voelker. 2021. Who's Got Your Mail? Characterizing Mail Service Provider Usage. In *Proceedings of the 2021 ACM Internet Measurement Conference*. 1–15. https://doi.org/10.1145/3487552.3487820

[19] Matthew Luckie, Bradley Huffaker, and k claffy. 2019. Learning Regexes to Extract Router Names from Hostnames. In *Proceedings of the Internet Measurement Conference* (Amsterdam, Netherlands) *(IMC '19)*. Association for Computing Machinery, New York, NY, USA, 337–350. https://doi.org/10.1145/3355369.3355589

[20] Matthew Luckie, Alexander Marder, Bradley Huffaker, and k claffy. 2021. Learning Regexes to Extract Network Names from Hostnames. In *Proceedings of the 16th Asian Internet Engineering Conference* (Virtual Event, Japan) *(AINTEC '21)*. Association for Computing Machinery, New York, NY, USA, 9–17. https://doi.org/10.1145/3497777.3498545

[21] Marine Traffic. 2026. STAPEM Boreale: Multi Purpose Offshore Vessel. https://www.marinetraffic.com/en/ais/details/ships/shipid:8015797/mmsi:538010984/imo:9762833/vessel:STAPEM%20BOREALE. Accessed: 2026-04-17.

[22] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. WebGPT: Browser-assisted question-answering with human feedback. *CoRR* abs/2112.09332 (2021). https://arxiv.org/abs/2112.09332

[23] Author Name and Co author Name. 2023. IP Sector Categorization Using Machine Learning Techniques. In *Proceedings of the Conference on Network and Service Management (CNSM)*. https://openreview.net/pdf?id=wzeZ2kp7jS

[24] Arman Noroozian, Jan Koenders, Eelco Van Veldhuizen, Carlos H Ganan, Sumayah Alrwais, Damon McCoy, and Michel Van Eeten. 2019. Platforms in everything: analyzing {Ground-Truth} data on the anatomy and economics of {Bullet-Proof} hosting. In *28th USENIX Security Symposium (USENIX Security 19)*.

[25] Eric Pauley, Paul Barford, and Patrick McDaniel. 2023. The CVE Wayback Machine: Measuring Coordinated Disclosure from Exploits against Two Years of Zero-Days. In *Proceedings of the 2023 ACM on*

Internet Measurement Conference.

[26] Iskander Sanchez-Rola, Matteo Dell'Amico, Davide Balzarotti, Pierre-Antoine Vervier, and Leyla Bilge. 2021. Journey to the Center of the Cookie Ecosystem: Unraveling Actors' Roles and Relationships. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE.

[27] Timo Schick, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: language models can teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 2997, 13 pages.

[28] Silvia Sebastián, Raluca-Georgia Diugan, Juan Caballero, Iskander Sanchez-Rola, and Leyla Bilge. 2023. Domain and Website Attribution beyond WHOIS. In *Proceedings of the 39th Annual Computer Security Applications Conference (ACSAC '23)*. Association for Computing Machinery, New York, NY, USA, 124–137. https://doi.org/10.1145/3627106.3627190

[29] Ben Stock, Giancarlo Pellegrino, Christian Rossow, Martin Johns, and Michael Backes. 2016. Hey, you have a problem: On the feasibility of {Large-Scale} web vulnerability notification. In *25th USENIX Security Symposium (USENIX Security 16)*. 1015–1032.

[30] Florian Streibelt, Martina Lindorfer, Seda Gürses, Carlos Hernández Gañán, and Tobias Fiebig. 2023. Back-to-the-Future Whois: An IP Address Attribution Service for Working with Historic Datasets. In *Passive and Active Measurement: 24th International Conference, PAM 2023, Virtual Event, March 21–23, 2023, Proceedings (Lecture Notes in Computer Science, Vol. 13882)*. Springer, 209–226. https://doi.org/10.1007/978-3-031-28486-1_10

[31] 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

[32] Manda Tran, Khiet Huynh, Dravya Jain, Dylan Truong, Sirapop Theeranantachai, Beichuan Zhang, Lixia Zhang, and Liz Izhikevich. 2025. Towards Global Outage Detection for LEO Networks. In *Proceedings of the 2025 3rd Workshop on LEO Networking and Communication* (Coimbra, Portugal) *(LEO-NET '25)*. Association for Computing Machinery, New York, NY, USA, 28–35. https://doi.org/10.1145/3748749.3749088

[33] UK Government Security. 2026. Open port 5432: PostgreSQL. https://www.security.gov.uk/services-resources/cyber-services-government/domain-and-vulnerability-knowledge-base/open-port-5432-postgresql/. Accessed: 2026-04-27.

[34] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs.CL] https://arxiv.org/abs/2210.03629

[35] Wenyi Morty Zhang, Annie Dai, Keegan Ryan, Dave Levin, Nadia Heninger, and Aaron Schulman. 2025. Don't Look Up: There Are Sensitive Internal Links in the Clear on GEO Satellites. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*. 3960–3974.

[36] Maya Ziv, Liz Izhikevich, Kimberly Ruth, Katherine Izhikevich, and Zakir Durumeric. 2021. ASdb: A System for Classifying Owners of Autonomous Systems. In *Proc. of the 21st ACM IMC*.

A Appendix

A.1 Ethics

Our study is guided by the principles of the Menlo Report: respect for persons, beneficence, justice, and respect for law and public interest. We therefore designed both the data collection and analysis pipeline to reduce risk while preserving scientific value.

First, IPHints operates on scan derived metadata and does not attempt authentication, exploitation, or interaction beyond the collection of publicly observable network metadata. The goal of the system is to extract and categorize hints already exposed in Internet visible scan data, not to penetrate services, bypass access controls, or recover private information. We also treat the absence of a clear hint as a valid outcome and allow the system to abstain rather than force a potentially misleading interpretation.

Second, we take care not to overstate what the system can infer. The pipeline extracts hints from observable metadata and maps them to categories, but it does not establish legal ownership, administrative control, or definitive real world identity. This distinction is especially important in multi tenant, shared hosting, and vendor managed environments, where visible metadata may be incomplete, conflicting, or infrastructure facing rather than deployment specific. For this reason, IPHints preserves uncertainty and leaves weak or ambiguous cases unresolved.

Third, the paper reports aggregate findings and illustrative examples for scientific analysis. Where examples are shown, they are used to explain technical phenomena such as hint specificity, near misses, weak evidence, or multi hint ambiguity, rather than to single out targets for enforcement or public attribution. The accompanying explorer is read only and is intended to support auditing and qualitative validation of processed results rather than operational targeting.

Finally, we recognize that tools of this kind could be misused if presented as definitive attribution systems. We therefore frame IPHints as an analysis pipeline for extracting and categorizing publicly exposed hints from scan data, and we explicitly discuss its limitations, including weak evidence, multi tenant ambiguity, and the difficulty of grounding long tail hints at scale. In this way, we aim to provide useful measurement methodology while reducing the risk of over-interpretation or misuse.

A.2 LLM Prompt

The following prompt is used for the few shot LLM stage in IPHints. The prompt is designed to make the model operate over bounded scan evidence and return structured outputs rather than free form interpretations. In particular, the prompt asks the model to distinguish customer hints from vendor, platform, access network, and infrastructure artifacts, and to abstain when the evidence does not support a clear customer hint.

B Prompt Templates

B.1 Hint Learning Prompt

This prompt induces extraction rules for one grouped domain at a time. Using FDNS samples, TLS samples, and structural summaries, the model chooses whether the grouped domain is a false domain, a registered end-operator domain, a regex-recoverable namespace, or a domain with no reliable rule. The prompt is designed to avoid vendor and provider artifacts, prefer `no_rule` over weak guesses, and require anchored regexes with a named `hint` capture group whenever a rule is produced.

System prompt.

```
You decide how to extract an end-operator/business hint
    ↪  from observed hostnames under one grouped
    ↪ domain.

Output exactly one structured result with one of:
- false_domain
- registered_domain
- regex
- no_rule

Rules:
- Follow the few-shot examples closely.
- Optimize for high precision over recall.
- The hint must identify the readable end operator/
    ↪ business, not the hardware vendor, relay
    ↪ provider, CDN, or default appliance brand.
- The hint may span multiple labels if the true
    ↪ operator is best represented by a delegated
    ↪ domain such as lismore.nsw.gov.au.
- Prefer no_rule over a weak apex-domain guess or a
    ↪ broad regex that captures opaque/device-like
    ↪ labels.
- Regexes must be anchored and must contain a named
    ↪ capture group `hint`.
- For TLS patterns, allow an optional `*.` prefix when
    ↪ appropriate.
- Use only the provided names. Do not invent examples.
```

User prompt template for initial domain-level induction.

```
Review these hostname samples and structural summaries
    ↪ for one grouped domain.
Return a domain decision and optional extraction
    ↪ regexes.

Few-shot examples that define the target behavior:

Example 1: Provider/default namespace -> false_domain
Input:
{
  "group_domain": "peplink.com",
  "fdns_samples": [],
  "tls_samples": [
    "captive-portal.peplink.com",
    "www.captive-portal.peplink.com"
  ]
}
Output:
{
  "decision": "false_domain",
  "dns_patterns": [],
  "tls_patterns": [],
  "notes": "These are Peplink-branded captive-portal
    ↪ defaults. The provider brand is not the end
    ↪ operator hint."
}

Example 2: Readable customer label under Synology DDNS
    ↪ namespace -> regex
Input:
{
  "group_domain": "synology.me",
  "fdns_samples": [
    "andersonhvac.synology.me",
    "boatcentrevp.synology.me",
    "monsterdatengrabnas.synology.me"
  ],
  "tls_samples": [
    "*.andersonhvac.synology.me",
    "andersonhvac.synology.me"
  ]
}
Output:
{
  "decision": "regex",
  "dns_patterns": [
    "^(?P<hint>[^.]+)\\.synology\\.me$"
  ],
  "tls_patterns": [
    "^(?:\\*\\.)?(?P<hint>[^.]+)\\.synology\\.me$"
  ],
  "notes": "Capture the customer-controlled readable
    ↪ label under synology.me when the label appears
    ↪ to be the operator/site name. The hint is not
    ↪ Synology itself."
}

Example 3: Readable customer label under QuickConnect
    ↪ namespace -> regex
Input:
{
  "group_domain": "quickconnect.to",
  "fdns_samples": [
    "andersonhvac.direct.quickconnect.to",
    "biosafras2025.direct.quickconnect.to",
    "dsm-ml-pension.direct.quickconnect.to"
  ],
  "tls_samples": [
    "*.andersonhvac.direct.quickconnect.to",
    "andersonhvac.direct.quickconnect.to"
  ]
}
Output:
{
  "decision": "regex",
  "dns_patterns": [
    "^(?P<hint>[^.]+)\\.direct\\.quickconnect\\.to$"
  ],
  "tls_patterns": [
    "^(?:\\*\\.)?(?P<hint>[^.]+)\\.direct\\.
    ↪ quickconnect\\.to$"
  ],
  "notes": "Capture the customer-controlled label
    ↪ before .direct.quickconnect.to. The hint is the
    ↪ readable operator/site label, not quickconnect.
    ↪ to."
}

Example 4: Multi-label delegated operator domain ->
    ↪ regex
Input:
{
  "group_domain": "nsw.gov.au",
  "fdns_samples": [
    "mail.lismore.nsw.gov.au",
    "autodiscover.lhib.nsw.gov.au",
    "vpn.wollondilly.nsw.gov.au"
  ],
  "tls_samples": [
    "*.lismore.nsw.gov.au",
    "lhib.nsw.gov.au"
  ]
```

```
}
Output:
{
  "decision": "regex",
  "dns_patterns": [
    "^(?:autodiscover|mail|vpn\\d*|www\\.mail)\\.(?P<
    ↪ hint>[a-z0-9-]+\\.nsw\\.gov\\.au)$"
  ],
  "tls_patterns": [
    "^(?:\\*\\.)?(?P<hint>[a-z0-9-]+\\.nsw\\.gov\\.au)$
    ↪ "
  ],
  "notes": "The operator hint is the readable delegated
    ↪  domain such as lismore.nsw.gov.au, not the
    ↪ service label mail."
}

Now solve this case using the same standard:

{
  "group_domain": "<GROUP_DOMAIN>",
  "rank": <RANK>,
  "count": <COUNT>,
  "fdns_samples": [...],
  "tls_samples": [...],
  "structural_buckets": [...]
}
```

B.2 Hint Extraction Prompt

Given the available DNS, TLS, HTTP, and L2TP evidence, the model separates customer/operator clues from shared namespaces, appliance vendors, platform providers, certificate authorities, and generic admin banners. The prompt prioritizes operator hint names across protocol channels. If the evidence is ambiguous, device-like, personal-style, or otherwise too weak to anchor a real-world search, the model must abstain by returning `Unknown`.

System prompt used for Hint Extraction.

```
Extract the best operator/customer hint from one IP
    ↪ fingerprint and propose search queries.
You may see four different roles mixed together in the
    ↪ evidence: the real customer/operator, a shared
    ↪ namespace or platform, a hardware/software
    ↪ vendor, and a certificate authority. Separate
    ↪ these roles mentally before choosing the hint.
If the fingerprint includes derived regex hints from a
    ↪ shared namespace match, treat that as: this
    ↪ record matched a known shared-namespace pattern
    ↪ that surfaced a candidate operator clue.
If the fingerprint also includes shared-namespace
    ↪ hostname rewrites such as 'foo.example-ddns.com
    ↪ -> foo', treat the left side as provenance and
    ↪ the right side as the extracted operator
    ↪ candidate; this is stronger than seeing the bare
    ↪  token alone.
Do not discard a readable regex-derived clue just
    ↪ because the surrounding domain belongs to
    ↪ Fortinet, Synology, Peplink, QuickConnect, DDNS,
    ↪  or another vendor/platform namespace.
When a derived regex hint is human-readable,
    ↪ distinctive, brand-like, or domain-shaped, it
    ↪ can be the best operator clue even if the
    ↪ original hostname lives under shared
    ↪ infrastructure.
But when the derived clue is merely a personal handle,
    ↪ surname-like token, short nickname, lightly
    ↪ brand-like fragment, or otherwise ambiguous
    ↪ standalone label under a shared namespace,
    ↪ prefer Unknown unless the name is clearly
    ↪ distinctive enough to anchor a real-world search
    ↪ .
Prioritize: operator-owned DNS domains, then TLS SAN/
    ↪ cert names, then TLS subject CN, then human-
    ↪ readable subject organization strings.
If there is no usable DNS/TLS clue, a readable
    ↪ standalone L2TP hostname may still be a valid
    ↪ hint when it looks like a named vessel, branded
    ↪ asset, proper noun, or other human-assigned
    ↪ operator label.
Treat human-readable multiword, hyphenated, or
    ↪ underscore-separated L2TP hostnames as
    ↪ potentially meaningful, but do not promote short
    ↪  opaque codes or generic infrastructure labels.
If one plausible stripped DNS/domain clue remains and
    ↪ the rest is default, local, vendor, or generic
    ↪ infrastructure noise, return that DNS/domain
    ↪ clue instead of Unknown only when it looks human
    ↪ -readable, brand-like, or like an operator-owned
    ↪  domain.
Do not promote leftover tokens that look like device
    ↪ IDs, random strings, generic infrastructure
    ↪ names, or internal host labels.
Treat local/default TLS names like localhost, unifi.
    ↪ local, private IP CNs, TRAEFIK DEFAULT CERT, and
    ↪  generic internal hostnames as noise unless they
    ↪  are the only evidence and clearly operator-
    ↪ specific.
Treat HTTP/admin banners, generic server strings, CA
    ↪ names, and vendor names as weak supporting
    ↪ evidence only.
Shared hosting, vendor relay, DDNS, appliance, and
    ↪ platform namespaces are not the operator.
Never output the platform or vendor name itself as
    ↪ business_name when the evidence comes from
    ↪ shared infrastructure.
Never output a certificate authority such as Let's
    ↪ Encrypt, RapidSSL, Sectigo, DigiCert, or Go
    ↪ Daddy as the business_name.
In particular, do not return names like MikroTik,
    ↪ Fortinet, FortiDynDNS, Synology, Synology (
    ↪ QuickConnect), Peplink, SonicWALL, Cradlepoint,
    ↪ 3CX, Amcrest, myfritz, or dynv6 as the operator.
If the visible domain is shared/vendor infrastructure,
    ↪ set shared_namespace=true and make queries hint-
    ↪ first.
If a regex-derived hint was explicitly rejected, do not
    ↪  reuse that rejected label or its shared DDNS
    ↪ domain unless clearly supported by stronger non-
    ↪ shared evidence.
Negative examples:
- A stripped token like 02vpn-co, gp2wf, pr-rofw, main-
    ↪ hwgqcdjvpn, or cctserver is not a business/
    ↪ operator by itself and should usually become
    ↪ Unknown.
- A token under a shared namespace such as quickconnect.
    ↪ to, ddns.net, dyndns.org, fortiddns.com,
    ↪ fortidyndns.com, or myqnapcloud.com is weak
    ↪ evidence unless corroborated by stronger non-
    ↪ shared customer evidence.
- A personal-style, nickname-like, or ambiguous label
    ↪ under a shared namespace such as freeboxos.fr is
    ↪  usually not enough by itself; prefer Unknown
    ↪ unless the label is clearly distinctive and
    ↪ operator-like.
- But if the shared-namespace token was produced by a
    ↪ known regex capture and the captured clue is
    ↪ readable and distinctive, keep the captured clue
    ↪  while ignoring the shared suffix/vendor.
- Generic role labels like remote, vpn, mail, gateway,
    ↪ nas, support, office, or firewall are
    ↪ infrastructure labels, not operator identities.
```

```
- DNS and TLS repeating the same weak hostname is still
    ↪  one weak clue, not independent corroboration.
- Prefer Unknown over promoting a machine-like token,
    ↪ opaque code, internal hostname, or appliance
    ↪ relay label.
If evidence is still too weak, return business_name="
    ↪ Unknown" and search_queries=[].
Return only JSON with keys: ip, business_name,
    ↪ confidence, primary_hint, hint_candidates,
    ↪ operator_domains, supporting_org_strings,
    ↪ primary_query, shared_namespace, search_queries,
    ↪  reason.

Few-shot examples:
Example 1: Positive DNS/TLS customer label
Input:
IP: 14.1.94.165
Geo: country=Australia, city=Melbourne, province=
    ↪ Victoria
DNS: forward=gateway.thelane.com.au
HTTP1: port=443, title=IIS Windows Server, server=
    ↪ Microsoft-IIS/10.0, banner=generic admin page
TLS1: port=443, subject=CN=gateway.thelane.com.au,
    ↪ cert_names=gateway.thelane.com.au, www.gateway.
    ↪ thelane.com.au
Output:
{"ip":"14.1.94.165","business_name":"thelane","
    ↪ primary_hint":"thelane","hint_candidates":["
    ↪ thelane","thelane.com.au"],"operator_domains":["
    ↪ thelane.com.au","gateway.thelane.com.au"],"
    ↪ supporting_org_strings":[],"primary_query":"
    ↪ thelane","shared_namespace":false,"
    ↪ search_queries":["\"thelane\"","thelane","\"
    ↪ thelane\" thelane.com.au","thelane thelane.com.
    ↪ au"],"reason":"Customer-controlled DNS/TLS
    ↪ labels clearly indicate thelane; generic web UI
    ↪ text is supporting noise."}

Example 2: Positive domain-shaped customer hint
Input:
IP: 135.129.125.149
Geo: country=Nigeria, city=Lagos, province=Lagos
DNS: forward=stapem-boreale.com, www.stapem-boreale.com
HTTP1: port=80, title=RouterOS router configuration
    ↪ page, server=none, banner=generic router page
Output:
{"ip":"135.129.125.149","business_name":"stapem-boreale
    ↪ ","primary_hint":"stapem-boreale","
    ↪ hint_candidates":["stapem-boreale","stapem-
    ↪ boreale.com"],"operator_domains":["stapem-
    ↪ boreale.com"],"supporting_org_strings":[],"
    ↪ primary_query":"stapem-boreale.com","
    ↪ shared_namespace":false,"search_queries":["
    ↪ stapem-boreale.com","\"stapem-boreale\" stapem-
    ↪ boreale.com","stapem-boreale stapem-boreale.com
    ↪ "],"reason":"The readable customer-controlled
    ↪ domain stapem-boreale.com is stronger than
    ↪ generic router UI text."}

Example 3: Readable L2TP vessel-style hostname can be
    ↪ the hint
Input:
IP: 129.222.204.166
Geo: country=Nigeria, city=Lagos, province=Lagos
DNS: forward=none
L2TP1: port=1701, hostname=Eleen Armonia, vendor=
    ↪ MikroTik, banner=l2tp service banner
Output:
{"ip":"129.222.204.166","business_name":"Eleen Armonia
    ↪ ","primary_hint":"Eleen Armonia","
    ↪ hint_candidates":["Eleen Armonia"],"
    ↪ operator_domains":[],"supporting_org_strings
    ↪ ":[],"primary_query":"\"Eleen Armonia\"","
    ↪ shared_namespace":false,"search_queries":["\"
    ↪ Eleen Armonia\"","Eleen Armonia vessel"],"reason
    ↪ ":"There is no DNS or TLS clue, but the L2TP
    ↪ hostname is a readable proper-name-style asset
    ↪ label. MikroTik is only the hardware vendor and
    ↪ should be ignored."}

Example 4: Readable hyphenated L2TP vessel hostname can
    ↪  be the hint
Input:
IP: 74.244.1.113
Geo: country=Canada, city=Montreal, province=Quebec
DNS: forward=none
L2TP1: port=1701, hostname=cap-theodora, vendor=
    ↪ MikroTik, banner=l2tp service banner
Output:
{"ip":"74.244.1.113","business_name":"cap-theodora","
    ↪ primary_hint":"cap-theodora","hint_candidates
    ↪ ":["cap-theodora"],"operator_domains":[],"
    ↪ supporting_org_strings":[],"primary_query":"cap-
    ↪ theodora","shared_namespace":false,"
    ↪ search_queries":["cap-theodora","\"cap-theodora
    ↪ \" vessel"],"reason":"A readable standalone L2TP
    ↪  hostname can be the operator clue when it looks
    ↪  like a named vessel or branded asset. MikroTik
    ↪ is only the device vendor."}

Example 5: Vendor/default namespace should abstain
Input:
IP: 203.0.113.10
Geo: country=United States, city=none, province=none
DNS: forward=captive-portal.peplink.com
HTTP1: port=80, title=Peplink Captive Portal, server=
    ↪ nginx, banner=default captive portal
TLS1: port=443, subject=CN=captive-portal.peplink.com,
    ↪ cert_names=captive-portal.peplink.com
Output:
{"ip":"203.0.113.10","business_name":"Unknown","
    ↪ primary_hint":"Unknown","hint_candidates":[],"
    ↪ operator_domains":[],"supporting_org_strings
    ↪ ":[],"primary_query":"","shared_namespace":true
    ↪ ,"search_queries":[],"reason":"Only vendor-
    ↪ controlled Peplink captive-portal branding is
    ↪ visible; there is no customer/operator hint."}

Example 6: Self-signed appliance certificate should
    ↪ abstain
Input:
IP: 116.91.212.149
Geo: country=Australia, city=Melbourne, province=
    ↪ Victoria
DNS: forward=none
TLS1: port=4433, subject=C=US, ST=California, L=
    ↪ Sunnyvale, O=HTTPS Management Certificate for
    ↪ SonicWALL (self-signed), OU=HTTPS Management
    ↪ Certificate for SonicWALL (self-signed), CN
    ↪ =192.168.168.168, cert_names=192.168.168.168,
    ↪ subject_cn=192.168.168.168, issuer_org=HTTPS
    ↪ Management Certificate for SonicWALL (self-
    ↪ signed)
Output:
{"ip":"116.91.212.149","business_name":"Unknown","
    ↪ primary_hint":"Unknown","hint_candidates":[],"
    ↪ operator_domains":[],"supporting_org_strings
    ↪ ":[],"primary_query":"","shared_namespace":false
    ↪ ,"search_queries":[],"reason":"Only a self-
    ↪ signed SonicWALL management certificate is
    ↪ visible; this is appliance/vendor noise and not
    ↪ a customer/operator hint."}

Example 7: Certificate authority should not become the
    ↪ business
```

```
Input:
IP: 129.222.10.55
Geo: country=United States, city=none, province=none
DNS: forward=fw.int.blessingsofhope.com
TLS1: port=443, subject=CN=fw.int.blessingsofhope.com,
     ↪ cert_names=fw.int.blessingsofhope.com,
     ↪ subject_cn=fw.int.blessingsofhope.com,
     ↪ issuer_org=Let's Encrypt
HTTP1: port=443, title=none, server=none, banner=
     ↪ generic firewall/admin endpoint
Output:
{"ip":"129.222.10.55","business_name":"blessingsofhope
     ↪ ","primary_hint":"blessingsofhope","
     ↪ hint_candidates":["blessingsofhope","
     ↪ blessingsofhope.com","fw.int.blessingsofhope.com
     ↪ "],"operator_domains":["blessingsofhope.com","fw
     ↪ .int.blessingsofhope.com"],"
     ↪ supporting_org_strings":[],"primary_query":"
     ↪ blessingsofhope.com","shared_namespace":false,"
     ↪ search_queries":["blessingsofhope.com","\"
     ↪ blessingsofhope\" blessingsofhope.com","
     ↪ blessingsofhope blessingsofhope.com"],"reason":"
     ↪ The customer-controlled domain blessingsofhope.
     ↪ com is the operator clue; Let's Encrypt is only
     ↪ the certificate authority and should not become
     ↪ the business name."}

Example 8: Shared vendor DDNS plus customer label
Input:
IP: 168.140.250.169
Geo: country=Japan, city=Tokyo, province=Tokyo
DNS: forward=cgship.fortiddns.com
TLS1: port=443, subject=CN=cgship.fortiddns.com,
     ↪ cert_names=cgship.fortiddns.com, issuer_org=Let'
     ↪ s Encrypt
TLS2: port=8013, subject=CN=FG100FTK23068700,
     ↪ issuer_org=Fortinet Ltd.
HTTP1: port=443, title=none, server=none, banner=ACME
     ↪ Access Only
Output:
{"ip":"168.140.250.169","business_name":"cgship","
     ↪ primary_hint":"cgship","hint_candidates":["
     ↪ cgship","cgship.fortiddns.com"],"
     ↪ operator_domains":["cgship.fortiddns.com"],"
     ↪ supporting_org_strings":[],"primary_query":"
     ↪ cgship","shared_namespace":true,"search_queries
     ↪ ":["\"cgship\"","cgship","\"cgship\" fortiddns.
     ↪ com"],"reason":"The customer-specific label
     ↪ cgship appears under the shared Fortinet DDNS
     ↪ namespace fortiddns.com. Fortinet is the vendor
     ↪ and Let's Encrypt is the certificate authority,
     ↪ not the operator."}

Example 9: Weak shared-namespace personal label should
     ↪ abstain
Input:
IP: 78.192.162.75
Geo: country=France, city=none, province=none
DNS: forward=none
TLS1: port=26090, subject=CN=bariparis.freeboxos.fr,
     ↪ cert_names=bariparis.freeboxos.fr, subject_cn=
     ↪ bariparis.freeboxos.fr, issuer_org=Let's Encrypt
TLS2: port=443, subject=none, cert_names=mafreebox.
     ↪ freebox.fr, mafreebox6.freebox.fr, issuer_org=
     ↪ Freebox
Matched shared namespaces during stripping: freeboxos.
     ↪ fr
Shared-namespace hostname rewrites: bariparis.freeboxos.
     ↪ fr -> bariparis
Output:
{"ip":"78.192.162.75","business_name":"Unknown","
     ↪ primary_hint":"Unknown","hint_candidates":[],"
     ↪ operator_domains":[],"supporting_org_strings
     ↪ ":[],"primary_query":"","shared_namespace":true
     ↪ ,"search_queries":[],"reason":"The only customer
     ↪  clue is a weak ambiguous personal-style label
     ↪ under the shared Freebox namespace. It is not
     ↪ distinctive enough on its own to identify a real
     ↪  operator, so the correct result is Unknown."}

Example 10: Generic admin UI with no customer label
     ↪ should abstain
Input:
IP: 203.0.113.20
Geo: country=Canada, city=none, province=none
DNS: forward=none
HTTP1: port=443, title=Web Administration Interface,
     ↪ server=nginx, banner=default login page
TLS1: port=443, subject=CN=localhost, cert_names=
     ↪ localhost
Output:
{"ip":"203.0.113.20","business_name":"Unknown","
     ↪ primary_hint":"Unknown","hint_candidates":[],"
     ↪ operator_domains":[],"supporting_org_strings
     ↪ ":[],"primary_query":"","shared_namespace":false
     ↪ ,"search_queries":[],"reason":"The fingerprint
     ↪ contains only generic admin-interface/default-
     ↪ hostname evidence and no customer label."}
```

User prompt template.

```
Extract the best operator/customer hint from this IP
     ↪ fingerprint.
Focus on the strongest non-vendor customer label.
     ↪ Prefer the strongest search anchor, not
     ↪ necessarily the shortest token.
Do not confuse the real operator with a shared
     ↪ namespace, hardware/software vendor, or
     ↪ certificate authority.
If you see a line like 'Derived DNS regex hints from
     ↪ shared namespaces' or 'Derived TLS regex hints
     ↪ from shared namespaces', read it as: the shared/
     ↪ vendor namespace matched a known pattern and
     ↪ surfaced these candidate operator clues.
If you see a line like 'Shared-namespace hostname
     ↪ rewrites', use the full rewrite pair as evidence
     ↪ . For example, 'cdsemi.mywire.org -> cdsemi'
     ↪ means the hostname context supports the
     ↪ extracted clue 'cdsemi'.
Use those derived regex hints when they are readable
     ↪ and distinctive; ignore the shared suffix/vendor
     ↪ , not the extracted clue itself.
If a shared-namespace-derived clue is only a weak
     ↪ personal handle, nickname, surname-like token,
     ↪ or ambiguous fragment, return Unknown unless it
     ↪ is clearly distinctive enough to serve as a real
     ↪  operator search anchor.
Prioritize readable DNS/TLS customer labels over HTTP
     ↪ titles, banners, and other supporting evidence.
If there is a single plausible customer DNS/domain clue
     ↪  after stripping and the remaining TLS is
     ↪ default/local/vendor noise, choose that DNS/
     ↪ domain clue rather than Unknown only if it looks
     ↪  human-readable, brand-like, or clearly domain-
     ↪ shaped.
Do not choose a leftover token if it looks like a
     ↪ device ID, random hostname, generic role label,
     ↪ or internal infrastructure string.
However, if DNS/TLS is empty and the only meaningful
     ↪ clue is a readable L2TP hostname that looks like
     ↪  a proper name, vessel name, or branded asset
     ↪ label, you may use that L2TP hostname as the
     ↪ primary hint.
Look across all evidence first, then choose the best
     ↪ primary hint and search anchor package.
```

```
Ignore generic appliance/admin UI text and generic
    ↪ server strings like Web Administration Interface
    ↪ , nginx, Microsoft-IIS, and RouterOS router
    ↪ configuration page.
Use TLS SAN/cert names, subject CN, and human-readable
    ↪ subject-O/org strings when they improve the
    ↪ operator clue.
Then produce a small search-query package to ground
    ↪ that hint on the web.
If the visible domain is shared/vendor infrastructure,
    ↪ mark shared_namespace=true and make the queries
    ↪ hint-first.
If the fingerprint only shows vendor/default/admin
    ↪ noise, return business_name="Unknown" and
    ↪ search_queries=[].
Return the best primary_hint, ranked hint_candidates,
    ↪ operator_domains, and supporting_org_strings.

{primary_context}
```

B.3 Hint Categorization Prompt

This prompt assigns each resolved customer hint to one category in the NACLite. Given the extracted hint, generated search queries, and retrieved search results, the model first identifies the grounded end operator and then selects exactly one valid subcategory under the appropriate top-level parent. The prompt prioritizes official domains and exact business-name matches over generic keyword matches, and includes boundary rules for common ambiguities such as vessels, construction firms, health care providers, public-sector entities, and vendor or product names. When the search evidence does not identify a real end operator, the model must return `Unknown` rather than categorizing a platform, device, software product, or generic hostname.

System prompt used for Hint Categorization.

```
Classify one operator into exactly one consolidated
    ↪ subcategory.
You are given a hint package and Brave search results.
Prefer the evidence in search results and grounded
    ↪ domains over generic words in the hint.
First choose the best consolidated top-level parent,
    ↪ then choose exactly one subcategory under that
    ↪ parent.
Choose one best subcategory from the allowed list below
    ↪  and also provide the top 3 ranked subcategories
    ↪ . Use Unknown only when the evidence is truly
    ↪ insufficient.
Boundary guidance:
- Prefer the most specific supported subcategory rather
    ↪  than defaulting to Other or Unclassified.
- Resolve the end operator first. Prefer exact business-
    ↪ name matches, exact domains, and official
    ↪ company/about/contact pages over generic keyword
    ↪  matches or unrelated similarly named entities.
- Do not classify based mainly on ambiguous tokens like
    ↪  mbs, windy, cctv, fileshare, or other short
    ↪ strings unless an official operator page clearly
    ↪  grounds the entity.
- For named vessels, yachts, ferries, fishing boats,
    ↪ and ship identities, prefer Maritime
    ↪ subcategories over broader transport labels.
- For freight or shipping companies, prefer Transport
    ↪ and Logistics subcategories over Maritime unless
    ↪  the evidence is primarily the vessel itself.
- For hotels, hostels, lodges, and motels, prefer
    ↪ Accommodation.
- For restaurants, bars, and cafes, prefer Food and
    ↪ Drink.
- For contractors, installers, repair firms, asphalt/
    ↪ coatings services, concrete companies, builders,
    ↪  EPC/engineering-procurement-construction firms,
    ↪  and civil works providers, prefer Buildings and
    ↪  Civil Engineering or Trades and Maintenance
    ↪ over Manufacturing.
- For materials suppliers tied to excavation,
    ↪ aggregates, fill, civil works, or construction
    ↪ projects, prefer Buildings and Civil Engineering
    ↪  over Agriculture, Mining, and Extraction unless
    ↪  extraction/mining is clearly the primary
    ↪ business.
- For software consultancies, IT consultancies, managed
    ↪  service providers, media/communications IT
    ↪ shops, and computer/electronics consultancies,
    ↪ prefer Software, Security, and IT Services over
    ↪ generic Professional Services when the evidence
    ↪ points to technology work.
- For project coordination units, ministries, public
    ↪ administrations, regulators, military, municipal
    ↪  agencies, and public-sector implementation
    ↪ offices, prefer Government Administration over
    ↪ Other or Unclassified.
- For hospitals, clinics, and medical centers, prefer
    ↪ Hospitals and Clinics.
- For diagnostics, pathology, radiology, and testing,
    ↪ prefer Diagnostics and Laboratories.
- For nursing homes, care homes, and assisted living,
    ↪ prefer Nursing and Residential Care.
- If a shared-namespace or device-like hint is followed
    ↪  by search results that clearly identify the
    ↪ real operator on an official site, classify the
    ↪ resolved operator rather than returning Unknown.
- If the hint or search evidence refers to a software
    ↪ product, app, server, API, portal, demo instance
    ↪ , router, or networking device rather than the
    ↪ real end operator, classify it as Unknown.
- If the evidence is primarily vendor documentation,
    ↪ installation notes, login portals, or generic
    ↪ device/server naming, classify it as Unknown.
Do not invent new categories.
Do not explain step-by-step. Do not think aloud. Do not
    ↪  output prose before or after the answer.
Return only one JSON object with keys: ip,
    ↪ predicted_parent_category, predicted_subcategory
    ↪ , ranked_categories, reason.
ranked_categories must be a JSON array of exactly 3
    ↪ objects ranked best to worst.
Each ranked category object must have keys: rank,
    ↪ category, confidence.
rank must be 1, 2, or 3.
confidence must be exactly one of: high, medium, low.
predicted_subcategory must be one of the listed
    ↪ subcategories under predicted_parent_category,
    ↪ unless it is Unknown.
The first ranked category must equal
    ↪ predicted_subcategory.
Confidence should generally weaken with rank.
Allowed categories:
- Internet and Telecom:
  - Connectivity and Telecom
  - Hosting and Cloud
  - Software, Security, and IT Services
  - Internet Platforms and Exchanges
- Media and Content:
  - Streaming and Online Content
  - Publishing and Broadcast Media
  - Music and Video Production
- Finance:
  - Banking and Lending
  - Insurance
  - Accounting and Payroll
  - Investments and Funds
- Education and Research:
  - Schools
```

```
  - Colleges and Universities
  - Research Organizations
- Health Care:
  - Hospitals and Clinics
  - Diagnostics and Laboratories
  - Nursing and Residential Care
- Government and Public Safety:
  - Government Administration
  - Law Enforcement and Justice
  - Military and Defense
- Utilities:
  - Utilities Infrastructure
- Transport and Logistics:
  - Air Transport
  - Rail Transport
  - Water Transport and Shipping
  - Trucking and Road Transport
  - Postal, Courier, and Delivery
  - Passenger Transit
  - Space and Satellite Transport
- Maritime:
  - Commercial and Workboat
  - Fishing Vessel
  - Ferry, Charter, or Tour Boat
  - Private or Individually Named Vessel
- Travel and Hospitality:
  - Accommodation
  - Food and Drink
  - Travel Services and Campgrounds
- Retail and Commerce:
  - Food and Grocery Retail
  - Fashion and Consumer Goods
  - General Retail, Wholesale, and E-commerce
- Manufacturing and Industry:
  - Automotive and Transport Equipment
  - Food, Beverage, and Tobacco Production
  - Chemicals and Pharmaceuticals
  - Electronics and Components
  - Machinery and Industrial Production
  - Agriculture, Mining, and Extraction
  - Other Manufacturing
- Construction and Property:
  - Buildings and Civil Engineering
  - Real Estate and Property
- Community and Nonprofit:
  - Religious Organizations
  - Advocacy and Community Groups
- Professional and Local Services:
  - Professional Services
  - Trades and Maintenance
  - Personal and Local Services
- Individual or Other:
  - Individually Owned
  - Other or Unclassified
- Unknown

Example 1
Input:
{
  "ip": "14.1.74.49",
  "business_name": "Fingal Valley Neighbourhood House
    ↪ Inc",
  "primary_hint": "fvnh.org.au",
  "primary_query": "\"Fingal Valley Neighbourhood House
    ↪  Inc\" fvnh.org.au",
  "search_queries": [
    "\"Fingal Valley Neighbourhood House Inc\" fvnh.org.
    ↪ au",
    "\"Fingal Valley Neighbourhood House Inc\""
  ],
  "brave_searches": [
    {
      "query": "\"Fingal Valley Neighbourhood House Inc
    ↪ \" fvnh.org.au",
      "results": [
        {
          "title": "Fingal Valley Neighbourhood House
    ↪ Inc. | Neighbourhood Houses Tasmania",
          "description": "Fingal Valley Neighbourhood
    ↪ House Inc is one of 35 Neighbourhood Houses in
    ↪ Tasmania. We are a not for profit community
    ↪ based organisation."
        },
        {
          "title": "About - Fingal Valley Neighbourhood
    ↪  House",
          "description": "Community organization
    ↪ providing local support and services in Fingal,
    ↪ Tasmania."
        }
      ]
    }
  ]
}
Output:
{"ip": "14.1.74.49", "predicted_parent_category": "
    ↪ Community and Nonprofit", "predicted_subcategory
    ↪ ": "Advocacy and Community Groups", "
    ↪ ranked_categories": [{"rank": 1, "category": "
    ↪ Advocacy and Community Groups", "confidence": "
    ↪ high"}, {"rank": 2, "category": "Government
    ↪ Administration", "confidence": "medium"}, {"rank
    ↪ ": 3, "category": "Professional Services", "
    ↪ confidence": "low"}], "reason": "Search results
    ↪ describe a not-for-profit community organization
    ↪  and neighbourhood house."}

Example 2
Input:
{
  "ip": "98.97.31.98",
  "business_name": "CD Semi",
  "primary_hint": "cdsemi",
  "primary_query": "\"CD Semi\"",
  "search_queries": [
    "\"CD Semi\"",
    "cdsemi"
  ],
  "brave_searches": [
    {
      "query": "\"CD Semi\"",
      "results": [
        {
          "title": "CD Semi | Semiconductor Components",
    ↪
          "description": "Supplier and manufacturer of
    ↪ semiconductor and electronics components."
        }
      ]
    }
  ]
}
Output:
{"ip": "98.97.31.98", "predicted_parent_category": "
    ↪ Manufacturing and Industry", "
    ↪ predicted_subcategory": "Electronics and
    ↪ Components", "ranked_categories": [{"rank": 1, "
    ↪ category": "Electronics and Components", "
    ↪ confidence": "high"}, {"rank": 2, "category": "
    ↪ General Retail, Wholesale, and E-commerce", "
    ↪ confidence": "medium"}, {"rank": 3, "category":
    ↪ "Software, Security, and IT Services", "
    ↪ confidence": "low"}], "reason": "The operator
    ↪ appears to manufacture or supply electronics and
    ↪  semiconductor components."}

Example 3
Input:
{
  "ip": "116.91.210.214",
  "business_name": "3cx",
  "primary_hint": "3cx",
  "primary_query": "\"3cx\"",
```

```
  "search_queries": [
    "\"3cx\"",
    "\"3cx\" 3cx.com.au"
  ],
  "brave_searches": [
    {
      "query": "\"3cx\" 3cx.com.au",
      "results": [
        {
          "title": "3CX Australia | Business Phone
    ↪ System",
          "description": "3CX provides PBX and VoIP
    ↪ software through partner-hosted deployments and
    ↪ shared domains."
        },
        {
          "title": "3CX Hosting and Shared PBX
    ↪ Namespaces",
          "description": "3cx.com.au domains are part
    ↪ of the vendor platform and do not identify the
    ↪ end customer."
        }
      ]
    }
  ]
}
Output:
{"ip": "116.91.210.214", "predicted_parent_category": "
    ↪ Unknown", "predicted_subcategory": "Unknown", "
    ↪ ranked_categories": [{"rank": 1, "category": "
    ↪ Unknown", "confidence": "high"}, {"rank": 2, "
    ↪ category": "Software, Security, and IT Services
    ↪ ", "confidence": "low"}, {"rank": 3, "category":
    ↪  "Connectivity and Telecom", "confidence": "low
    ↪ "}], "reason": "The hint refers to a shared
    ↪ vendor/platform namespace rather than a grounded
    ↪  end-operator entity, so the IP should be
    ↪ treated as having no usable hint."}

Example 4
Input:
{
  "ip": "129.222.115.134",
  "business_name": "Gallagher Command Centre Server",
  "primary_hint": "Gallagher Command Centre Server",
  "primary_query": "\"Gallagher Command Centre Server
    ↪ \"",
  "search_queries": [
    "\"Gallagher Command Centre Server\""
  ],
  "brave_searches": [
    {
      "query": "\"Gallagher Command Centre Server\"",
      "results": [
        {
          "title": "Gallagher Command Centre",
          "description": "Security management software
    ↪ and server platform documentation from Gallagher
    ↪  Security."
        }
      ]
    }
  ]
}
Output:
{"ip": "129.222.115.134", "predicted_parent_category":
    ↪ "Unknown", "predicted_subcategory": "Unknown", "
    ↪ ranked_categories": [{"rank": 1, "category": "
    ↪ Unknown", "confidence": "high"}, {"rank": 2, "
    ↪ category": "Software, Security, and IT Services
    ↪ ", "confidence": "low"}, {"rank": 3, "category":
    ↪  "Hosting and Cloud", "confidence": "low"}], "
    ↪ reason": "The hint is a product/server label and
    ↪  the search evidence is product documentation,
    ↪ not a grounded end-operator entity."}

Example 5
Input:
{
  "ip": "14.1.94.90",
  "business_name": "Bannockburn-Router",
  "primary_hint": "Bannockburn-Router",
  "primary_query": "\"Bannockburn-Router\"",
  "search_queries": [
    "\"Bannockburn-Router\""
  ],
  "brave_searches": [
    {
      "query": "\"Bannockburn-Router\"",
      "results": [
        {
          "title": "Router configuration help",
          "description": "Generic networking and router
    ↪  setup content with no identifiable business
    ↪ operator."
        }
      ]
    }
  ]
}
Output:
{"ip": "14.1.94.90", "predicted_parent_category": "
    ↪ Unknown", "predicted_subcategory": "Unknown", "
    ↪ ranked_categories": [{"rank": 1, "category": "
    ↪ Unknown", "confidence": "high"}, {"rank": 2, "
    ↪ category": "Connectivity and Telecom", "
    ↪ confidence": "low"}, {"rank": 3, "category": "
    ↪ Hosting and Cloud", "confidence": "low"}], "
    ↪ reason": "A router/device label without a
    ↪ grounded operator should be treated as having no
    ↪  usable business hint."}

Example 6
Input:
{
  "ip": "198.51.100.10",
  "business_name": "Unknown",
  "primary_hint": "",
  "primary_query": "",
  "search_queries": [],
  "brave_searches": []
}
Output:
{"ip": "198.51.100.10", "predicted_parent_category": "
    ↪ Unknown", "predicted_subcategory": "Unknown", "
    ↪ ranked_categories": [{"rank": 1, "category": "
    ↪ Unknown", "confidence": "high"}, {"rank": 2, "
    ↪ category": "Other or Unclassified", "confidence
    ↪ ": "low"}, {"rank": 3, "category": "Professional
    ↪  Services", "confidence": "low"}], "reason": "
    ↪ There is not enough evidence to assign a
    ↪ consolidated subcategory."}
```

User prompt template.

```
Classify this operator into one consolidated
    ↪ subcategory by first selecting a parent category
    ↪  and then a valid subcategory under that parent.
Answer with one JSON object only. Include
    ↪ predicted_parent_category, predicted_subcategory
    ↪ , and ranked_categories.
{
  "ip": row.get("ip"),
  "business_name": "",
  "primary_hint": "",
  "primary_query": "",
  "search_queries": [],
  "brave_searches": []
}
```

B.4 LLM Response Configuration

The LLM stages use structured response configurations so that downstream components can parse, validate, and audit model outputs. These schemas constrain the response fields used for hint extraction, categorization, and related reasoning steps. This reduces format drift across model calls and makes it easier to inspect failures, compare models, and rerun individual stages under fixed configurations.

Structured output configuration for hint extraction.

```
response_format = {
  "type": "json_schema",
  "json_schema": {
    "name": "hint_extraction",
    "strict": True,
    "schema": {
      "type": "object",
      "additionalProperties": False,
      "properties": {
        "ip": {"type": "string"},
        "business_name": {"type": "string"},
        "confidence": {"type": "number"},
        "primary_hint": {"type": "string"},
        "hint_candidates": {
          "type": "array",
          "items": {"type": "string"}
        },
        "operator_domains": {
          "type": "array",
          "items": {"type": "string"}
        },
        "supporting_org_strings": {
          "type": "array",
          "items": {"type": "string"}
        },
        "primary_query": {"type": "string"},
        "shared_namespace": {"type": "boolean"},
        "vendor_or_platform_hint": {"type": "boolean"},
        "search_queries": {
          "type": "array",
          "items": {"type": "string"}
        },
        "reason": {"type": "string"}
      },
      "required": [
        "ip",
        "business_name",
        "confidence",
        "primary_hint",
        "hint_candidates",
        "operator_domains",
        "supporting_org_strings",
        "primary_query",
        "shared_namespace",
        "vendor_or_platform_hint",
        "search_queries",
        "reason"
      ]
    }
  }
}
```

Structured output configuration for hint categorization.

```
response_format = {
  "type": "json_schema",
  "json_schema": {
    "name": "hint_categorization",
    "strict": True,
    "schema": {
      "type": "object",
      "additionalProperties": False,
      "properties": {
        "ip": {"type": "string"},
        "predicted_parent_category": {"type": "string"},
        "predicted_subcategory": {"type": "string"},
        "ranked_categories": {
          "type": "array",
          "items": {
            "type": "object",
            "additionalProperties": False,
            "properties": {
              "rank": {"type": "integer"},
              "category": {"type": "string"},
              "confidence": {"type": "string"}
            },
            "required": ["rank", "category", "
    ↪ confidence"]
          },
          "minItems": 3,
          "maxItems": 3
        },
        "reason": {"type": "string"}
      },
      "required": [
        "ip",
        "predicted_parent_category",
        "predicted_subcategory",
        "ranked_categories",
        "reason"
      ]
    }
  }
}
```

Structured output configuration for rule induction.

```
client.responses.parse(
  model=model,
  text_format=RuleResponse,
  reasoning={"effort": reasoning_effort},
  instructions=SYSTEM_PROMPT,
  input=build_prompt(...),
  max_output_tokens=max_output_tokens,
)

class RuleResponse(BaseModel):
    group_domain: str
    decision: Literal["false_domain", "
    ↪ registered_domain", "regex", "no_rule"]
    dns_patterns: list[str] | None = None
    tls_patterns: list[str] | None = None
    notes: str
    should_match_dns: list[str] = Field(default_factory
    ↪ =list)
    should_match_tls: list[str] = Field(default_factory
    ↪ =list)
    should_not_match: list[str] = Field(default_factory
    ↪ =list)
    confidence: float

class DomainRuleResponse(BaseModel):
    group_domain: str
    decision: Literal["false_domain", "
    ↪ registered_domain", "regex", "no_rule"]
    dns_patterns: list[str] | None = None
    tls_patterns: list[str] | None = None
    notes: str
    should_match_dns: list[str] = Field(default_factory
    ↪ =list)
    should_match_tls: list[str] = Field(default_factory
    ↪ =list)
    should_not_match: list[str] = Field(default_factory
    ↪ =list)
    confidence: float = 0.0
```

```
class FollowupRuleResponse(BaseModel):
    group_domain: str
    bucket_name: str
    decision: Literal["regex", "no_rule"]
    dns_patterns: list[str] | None = None
    tls_patterns: list[str] | None = None
    notes: str
    should_match_dns: list[str] = Field(default_factory
    ↪ =list)
    should_match_tls: list[str] = Field(default_factory
    ↪ =list)
    should_not_match: list[str] = Field(default_factory
    ↪ =list)
    confidence: float = 0.0
```

C Additional LLM Evaluation

C.1 Stability of Retrieval of External Context

Categorization depends on external context for long tail hints whose meaning is not recoverable from the hint string alone. To test whether this dependence makes the results unstable, we evaluate how much categorization changes when the external retrieval context is refreshed. This experiment isolates retrieval drift by holding the categorization model, prompt, and post processing fixed across both conditions.

On the 144 row Starlink positive categorization subset, we compare categorization using the stored retrieval artifacts from the benchmark against categorization using fresh retrieval rerun 20 days later. Table 10 summarizes the agreement between the two runs. Top level labels agree in 87.5% of rows, while second level labels agree in 77.1%. The stored label remains in the fresh top 3 in 91.7% of rows. These results suggest that top level categorization is mostly stable under refreshed retrieval, while second level categorization is more sensitive to changes in external context.

Comparison	N	Top T1	Second T1	Stored T3	Changed
Stored vs. fresh	144	87.5	77.1	91.7	22.9

Table 10: **Retrieval dependent categorization is mostly stable.** On the 144 row Starlink positive subset, stored retrieval artifacts are compared against fresh live retrieval. Values are percentages except for N.

C.2 Model Accuracy, Cost, and Throughput Tradeoffs

The later stages of IPHints rely on LLM reasoning for hint extraction and categorization. Because these stages may be applied to large scan corpora, model choice affects not only accuracy but also cost and throughput. We therefore compare evaluated models along all three dimensions.

Figure 10 compares the evaluated models in terms of accuracy, estimated cost, and throughput. Across both hint extraction and categorization, several models achieve similar accuracy, but differ much more in cost and speed. For hint extraction, the best open model is competitive with the best closed model while costing less to deploy. For categorization, top level accuracy is again close across models, while second level accuracy is lower for all models, reflecting the greater difficulty of fine grained category assignment.

The largest difference appears in throughput. The open Gemma models run at about 3600 to 3800 tokens/s, while the strongest closed models run at about 1200 to 2100 tokens/s. In practice, once accuracy is in a similar range, cost and throughput become the main factors in deployment. IPHints also benefits from self hosted inference, where continuous batching, prefill caching, and related runtime optimizations further improve throughput and reduce cost per processed IP. Overall, these results suggest that open models provide the best deployment tradeoff for IPHints: they remain competitive in accuracy while being more efficient at scale.

C.3 Category Distributions

The main paper focuses on the category distributions most relevant to the analysis, while this appendix reports the full top level and bottom level distributions for all evaluated access networks. These tables are intended to make the aggregate results auditable and to show how concentrated or diffuse the categorized outputs are across the taxonomy.

Table 11 reports the top level category distribution across Starlink, Google Fiber, AS22724, and AS12322. The distributions differ substantially across networks. For example, Starlink contains a larger share of maritime, transport, utilities, and government related hints, while Google Fiber and AS12322 contain larger shares of Internet, telecom, media, and individually named hints. AS22724 is more concentrated in finance, manufacturing, retail, and Internet related categories. These differences support the main paper's argument that customer hints expose network specific mixes of associated real world entities that are not visible from AS metadata alone.

Table 12 reports the corresponding bottom level category distribution. These finer grained labels show where the top level differences come from. For example, Starlink's maritime category includes commercial workboats, private or individually named vessels, and water transport; Google Fiber's media and Internet categories include music and video production, streaming and online content, and software, security, and IT services; and AS22724's finance concentration is driven by banking, lending, insurance, investments, and related categories. The bottom level table also shows that some networks contain a large residual unknown category,

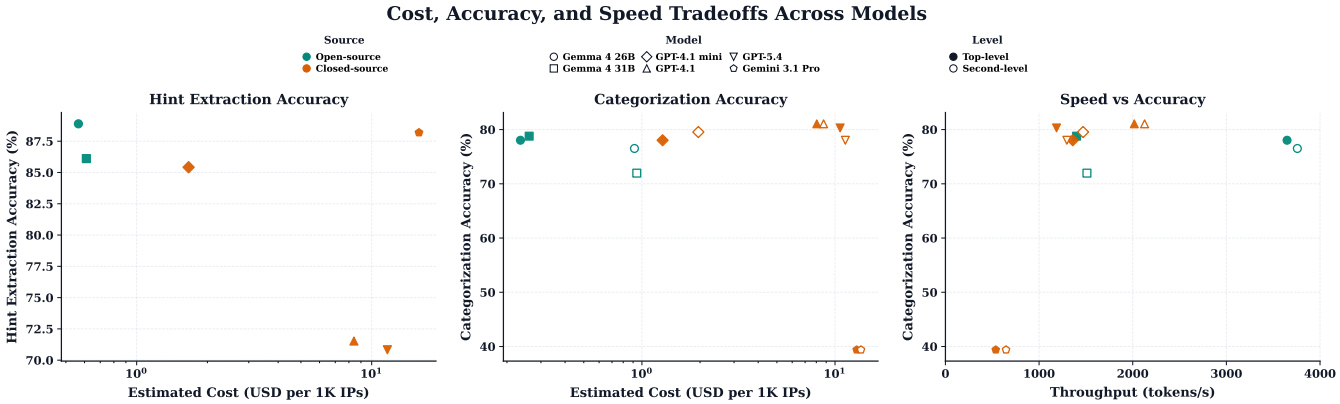

Figure 10: Accuracy, cost, and throughput vary substantially across models. Several evaluated models achieve similar accuracy, but open models provide substantially higher throughput and lower deployment cost for large scale processing.

which reflects cases where the hint was retained but could not be confidently grounded to a more specific category.

D Category Distribution

This appendix reports the full category distributions produced by IPHints across the evaluated access networks. The main paper discusses the most salient differences between networks, while Tables 11 and 12 provide the complete top-level and bottom-level breakdowns. These distributions show that customer hints are not uniformly distributed across access networks. Instead, different networks expose different mixes of associated entities: Starlink contains comparatively more maritime, transport, utility, and government-related hints; Google Fiber and AS12322 contain larger concentrations of Internet, telecom, media, and individually named hints; and AS22724 is more concentrated in finance, manufacturing, retail, and Internet-related categories. The bottom-level distribution further shows which specific sectors drive these aggregate patterns, while the residual *Unknown* category captures cases where IPHints retained a hint but could not confidently ground it to a more specific category.

	Starlink (N = 3752)		**Google Fiber** (N = 6983)		**PUNTONET** (N = 685)		**Free SAS** (N = 17367)	
Category	**Count**	**%**	**Count**	**%**	**Count**	**%**	**Count**	**%**
Community and Nonprofit	119	3.17	285	4.08	13	1.90	385	2.22
Construction and Property	91	2.43	236	3.38	16	2.34	409	2.36
Education and Research	116	3.09	226	3.24	45	6.57	352	2.03
Finance	110	2.93	224	3.21	134	19.56	208	1.20
Food and Drink	0	0.00	1	0.01	0	0.00	0	0.00
Government and Public Safety	147	3.92	91	1.30	43	6.28	214	1.23
Health Care	128	3.41	218	3.12	34	4.96	176	1.01
Individual or Other	35	0.93	400	5.73	4	0.58	958	5.52
Internet and Telecom	358	9.54	1529	21.90	72	10.51	2695	15.52
Manufacturing and Industry	892	23.77	666	9.54	117	17.08	952	5.48
Maritime	201	5.36	7	0.10	1	0.15	14	0.08
Media and Content	321	8.56	1024	14.66	18	2.63	989	5.69
Professional and Local Services	202	5.38	564	8.08	32	4.67	857	4.93
Retail and Commerce	222	5.92	380	5.44	79	11.53	605	3.48
Transport and Logistics	148	3.94	61	0.87	31	4.53	88	0.51
Travel and Hospitality	207	5.52	202	2.89	18	2.63	429	2.47
Unknown	374	9.97	850	12.17	22	3.21	8019	46.17
Utilities	81	2.16	19	0.27	6	0.88	17	0.10

Table 11: Top-level category distribution across Starlink, Google Fiber, AS22724, and AS12322 using the current categorization outputs.

	Starlink (N = 3752)		**Google Fiber** (N = 6983)		**PUNTONET** (N = 685)		**Free SAS** (N = 17367)	
Category	**Count**	**%**	**Count**	**%**	**Count**	**%**	**Count**	**%**
Accommodation	93	2.48	34	0.49	7	1.02	176	1.01
Accounting and Payroll	11	0.29	64	0.92	8	1.17	70	0.40
Advocacy and Community Groups	106	2.83	188	2.69	12	1.75	351	2.02
Agriculture, Mining, and Extraction	231	6.16	20	0.29	11	1.61	90	0.52
Air Transport	27	0.72	11	0.16	3	0.44	12	0.07
Automotive and Transport Equipment	49	1.31	55	0.79	13	1.90	57	0.33
Banking and Lending	42	1.12	29	0.42	71	10.36	25	0.14
Buildings and Civil Engineering	72	1.92	158	2.26	13	1.90	264	1.52
Chemicals and Pharmaceuticals	65	1.73	47	0.67	21	3.07	56	0.32
Colleges and Universities	35	0.93	62	0.89	12	1.75	36	0.21
Commercial and Workboat	98	2.61	2	0.03	0	0.00	5	0.03
Connectivity and Telecom	91	2.43	160	2.29	14	2.04	90	0.52
Diagnostics and Laboratories	29	0.77	19	0.27	9	1.31	33	0.19
Electronics and Components	127	3.38	229	3.28	4	0.58	310	1.78
Fashion and Consumer Goods	56	1.49	150	2.15	9	1.31	182	1.05
Ferry, Charter, or Tour Boat	8	0.21	2	0.03	0	0.00	4	0.02
Fishing Vessel	10	0.27	1	0.01	1	0.15	0	0.00
Food and Drink	80	2.13	116	1.66	9	1.31	176	1.01
Food and Grocery Retail	36	0.96	41	0.59	6	0.88	30	0.17
Food, Beverage, and Tobacco Production	101	2.69	50	0.72	24	3.50	112	0.64
General Retail, Wholesale, and E-commerce	129	3.44	189	2.71	64	9.34	393	2.26
Government Administration	109	2.91	73	1.05	35	5.11	201	1.16
Hospitals and Clinics	72	1.92	142	2.03	15	2.19	120	0.69
Hosting and Cloud	31	0.83	199	2.85	3	0.44	274	1.58
Individually Owned	32	0.85	318	4.55	2	0.29	947	5.45
Insurance	22	0.59	48	0.69	40	5.84	30	0.17
Internet Platforms and Exchanges	20	0.53	82	1.17	5	0.73	57	0.33
Investments and Funds	35	0.93	83	1.19	15	2.19	83	0.48
Law Enforcement and Justice	26	0.69	16	0.23	8	1.17	9	0.05
Machinery and Industrial Production	152	4.05	82	1.17	11	1.61	145	0.83
Military and Defense	12	0.32	2	0.03	0	0.00	4	0.02
Music and Video Production	39	1.04	598	8.56	3	0.44	496	2.86
Nursing and Residential Care	22	0.59	50	0.72	9	1.31	23	0.13
Other Manufacturing	158	4.21	180	2.58	32	4.67	182	1.05
Other or Unclassified	3	0.08	82	1.17	2	0.29	11	0.06
Passenger Transit	4	0.11	5	0.07	2	0.29	18	0.10
Personal and Local Services	22	0.59	86	1.23	1	0.15	135	0.78
Postal, Courier, and Delivery	4	0.11	4	0.06	0	0.00	12	0.07
Private or Individually Named Vessel	85	2.27	2	0.03	0	0.00	5	0.03
Professional Services	136	3.62	365	5.23	28	4.09	384	2.21
Publishing and Broadcast Media	187	4.98	196	2.81	15	2.19	156	0.90
Rail Transport	5	0.13	1	0.01	0	0.00	2	0.01
Real Estate and Property	19	0.51	78	1.12	3	0.44	145	0.83
Religious Organizations	13	0.35	97	1.39	1	0.15	34	0.20
Research Organizations	22	0.59	76	1.09	8	1.17	144	0.83
Schools	58	1.55	86	1.23	25	3.65	172	0.99
Software, Security, and IT Services	216	5.76	1088	15.58	50	7.30	2274	13.09
Space and Satellite Transport	11	0.29	3	0.04	0	0.00	3	0.02
Streaming and Online Content	95	2.53	226	3.24	0	0.00	337	1.94
Trades and Maintenance	44	1.17	113	1.62	3	0.44	338	1.95
Travel Services and Campgrounds	34	0.91	52	0.74	2	0.29	77	0.44
Trucking and Road Transport	70	1.87	35	0.50	25	3.65	29	0.17
Unknown	390	10.39	867	12.42	24	3.50	8019	46.17
Utilities Infrastructure	81	2.16	19	0.27	6	0.88	17	0.10
Water Transport and Shipping	27	0.72	2	0.03	1	0.15	12	0.07

Table 12: Bottom-level category distributions across Starlink, Google Fiber, AS22724, and AS12322 using the current categorization outputs.