Discovering Obscure Looking Glass Sites on the Web to Facilitate Internet Measurement Research

Shuying Zhuang
Tsinghua University

Jessie Hui Wang*
Tsinghua University

Jilong Wang
Tsinghua University

Zujiang Pan
Tencent Technology

Tianhao Wu, Fenghua Li
Tsinghua University

Zhiyong Zhang
CETC

ABSTRACT

Despite researchers have noticed that Looking Glass (LG) vantage points (VPs) are valuable for Internet measurement researches, they can only exploit VPs from well-known LG sites published on several LG portal pages. There should be a lot of LG sites that are not published in these portal pages, namely *obscure LG sites*, which are not easy to be found and exploited by researchers. In this paper, we design an efficient focused crawler to discover as many LG sites as possible which can avoid unnecessary resource consumption on analyzing irrelevant pages. Our designed focused crawler takes a similarity-guided search that exploits the well-developed search engines and comprehensively mines the common features shared by known LG sites to discover more LG pages. Moreover, the focused crawler takes a two-step PU learning classifier based on carefully selected LG features to efficiently discard irrelevant URLs, thus avoiding a lot of unnecessary resource consumption. As far as we know, we are the first to develop a method to discover obscure LG sites on the web. Experimental results show the effectiveness of our focused crawler. To facilitate practical applications, we further develop an automation tool, which can successfully retrieve 910 obscure automatable LG VPs from relevant pages obtained through our focused crawler. The 910 LG VPs significantly increase the geographic and network coverage of available VPs and we show their potential values in improving the completeness of AS-level Internet topology by a simple case study. Our method and the final VP list (which will be publicly accessible) are beneficial to the measurement community.

CCS CONCEPTS

• **Networks → Network measurement**.

KEYWORDS

Looking Glass, Internet Measurement, Focused Crawler

1 INTRODUCTION

Measurement vantage points (VPs) are critical for Internet measurement researches. Taking the inter-domain topology measurement as an example, the large-scale and distributed nature of the Internet implies the completeness of collected topologies heavily depends on the number and distribution of VPs. For years, measurement platforms (such as Archipelago, RouteViews) have spent a lot of efforts in deploying VPs. For example, the RouteViews platform has taken about 20 years in seeking the cooperation of organizations that are willing to share their routing information. However, the number and the geographic and network coverage of its VPs are still limited [21, 31, 32, 35, 51]. It would be very valuable if researchers can measure the Internet from more VPs.

Many Internet service providers actively deploy *Looking Glass* (LG) VPs inside their networks and allow users to run some popular measurement commands on these VPs, such as traceroute or bgp [29]. They deploy LGs to provide windows to observe their networks to attract customers to use their network services and help troubleshoot Internet connectivity and performance issues. These LG VPs offer researchers opportunities to observe the Internet from various locations, such as core routers, border routers, and IXPs' route servers.

Researchers usually find usable LG sites from several well-known LG portal pages, *i.e.*, PeeringDB [8], Traceroute.org [14], BGP4.as [2], and BGPLookingglass.com [3]. We name the LG sites published on these several portal pages as *well-known LG sites*, and VPs from well-known LG sites have been used in many measurement researches [37, 46, 47].

However, many LG sites that are not published on these well-known LG portal pages cannot be found and exploited easily. We refer to these LG sites as *obscure LG sites*. It would be valuable if researchers can exploit these obscure LG sites. In this paper, we design an efficient focused crawler to discover obscure LG sites on the web. To help researchers to use them efficiently, we further develop a tool to automate the use of their VPs and retrieve a list of automatable LG VPs. As far as we know, we are the first to develop a method to discover obscure LG sites on the web.

*Jessie Hui Wang is the corresponding author, jessiewang@tsinghua.edu.cn.

With the rapid growth of network information, there are near two billion active websites on the web. Instead of crawling every page and judge whether it provides a looking glass service to discover LG sites, the basic idea of our efficient focused crawler is to only crawl pages that are likely to be LG pages and judge whether they are LG pages. There are two key procedures, i.e. crawling procedure and classification procedure, that have to be carefully designed with consideration of the characteristics of LG sites, to achieve the goal of discovering as many LG sites as possible while avoiding unnecessary resource consumption on irrelevant pages.

Previous focused crawlers for various topics often assume that pages on the same topic are usually connected, which does not hold for LG pages. There can be many LG pages that are not connected to any other LG pages. As a result, the widely used hyperlink-guided crawling procedure is not sufficient for crawling pages that are likely to be LG pages. We need to design a new search method based on the characteristics of LG pages to efficiently discover more LG pages. In this work, we develop a similarity-guided search that exploits well-developed search engines to find pages that are similar to known LG pages. Search engines have indexed a complete view of the web and designed professional search algorithms. On this basis, we carefully analyze all well-known LG pages and extract their common features as search terms to get high-quality search results.

To efficiently and correctly judge URLs that are obtained from the crawling procedure, we design a two-step PU learning classifier based on carefully selected LG features. We notice that URLs, page titles, and specific elements in html files are all valuable for distinguishing between LG pages and other pages. So we first design a pre-filter based on URL features to filter out a lot of irrelevant URLs, thus avoiding significant resource consumption in downloading html files of the irrelevant URLs. Then we design a classifier based on carefully selected html file features to further classify the remaining URLs with downloaded html files more accurately. Moreover, the semi-supervised PU (Positive and Unlabeled) learning approach can help us deal with the problem of lacking labeled negative samples for training without requiring time-consuming work to manually label numerous samples.

Experimental results show the focused crawler can efficiently find more LG pages with lower resource overhead. Specifically, our similarity-guided search can discover relevant pages 24 times more than those discovered from the widely used hyperlink-based search. Our two-step classifier would be both accurate and efficient. It can not only effectively reduce the number of URLs to be downloaded by about 85%, but also obtain final classification results with high TPR (96.10%) and low FPR (4.10%), which means most LG pages are reserved as relevant ones and most non-LG pages are filtered out. In total, our focused crawler discovers 50,777 pages that are relevant to LG sites.

Despite the pages that are classified as relevant by the focused crawler are not guaranteed to be LG pages, the probability for a relevant page to be an LG page has been much higher than a random page. Therefore these relevant pages are useful and researchers can check and use them manually when they need more VPs. As for researchers who want to conduct large-scale or periodic measurement tasks, it will be more useful if they can use the VPs on these LG pages by just sending requests automatically. We develop a tool to automate the use of LG VPs and retrieve 910 obscure automatable VPs from the relevant pages. As a comparison, there are only 1,446 known automatable VPs from all well-known LG sites. Our work increases the number of automatable LG VPs that are available to researchers by about 62.9%.

The 910 obscure VPs enable researchers to execute measurement commands from 8 new countries, 160 new cities, and 262 new ASes where no known VPs (including measurement platforms such as Ark and RIPE) have been found before. To show their potential values in facilitating Internet measurement research, we conduct a case study in which we use obscure LG VPs to improve the completeness of the AS-level Internet topology. In this case study, we use only 8 obscure automatable VPs that support both commands of *show ip bgp summary* and *show bgp neighbor ip advertised (or received) routes*. These 8 VPs help us find 1,428 new AS links and 10 new ASes that are not observed by VPs from well-known LG sites, RIPE RIS, and RouteViews, which are almost all public VP sources that can be used to obtain BGP routes before our work.

The rest of this article is structured as follows. In §2, we present the detailed design of our focused crawler. Experimental and evaluation results on our designed similarity-guided search and two-step classifier are presented in §3. We introduce the practical applications of our work in §4. In §5 and 6, we introduce ethical concerns and the related work respectively. §7 concludes the paper.

2 LG FOCUSED CRAWLER DESIGN

2.1 Design goals and challenges

There are extremely large amounts of pages on the web. Obviously, due to resource limitations, we cannot crawl every page and judge whether the page provides a looking glass service to discover LG pages. The goal of our work is to discover as many LG pages as possible while avoiding unnecessary resource consumption on irrelevant pages. However, building an LG focused crawler system that can achieve the goal is challenging for several reasons. First, the poor hyperlink connections between LG pages require us to design an effective search method to locate more LG pages. Second, to judge crawled pages efficiently and correctly, we need to design a high-performance classifier based on carefully selected LG features. Moreover, the lack of labeled non-LG pages (negative samples) further increases the difficulty of classifier design.

2.2 Overview

Figure 1 shows an overview of our LG focused crawler, which consists of two key components: a crawling procedure which searches the Internet for candidate URLs (§2.3) and a classification procedure which classifies candidate URLs into relevant or not (§2.4). It starts from a set of well-known LG sites, which is referred to as *LG seed set*. Based on these seed pages, our crawling procedure searches the Internet for URLs that are likely to be LG URLs, which are denoted as *candidate URLs*. We implement the search in two parallel ways, one is collecting hyperlinks of seed pages (namely *hyperlink-guided search*) and the other is exploiting well-developed search engines to collect pages similar to seed pages (namely *similarity-guided search*). Our similarity-guided search benefits from a comprehensive extraction of common features shared by known LG pages thus can help locate more effective candidate URLs.

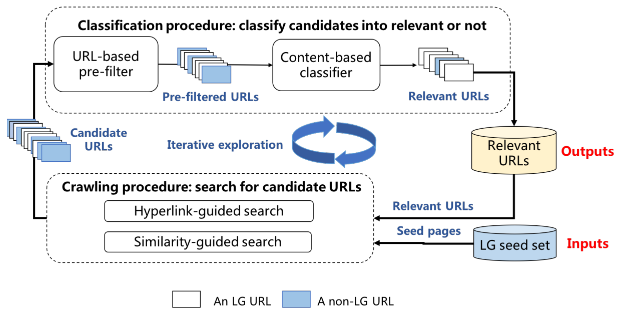

Figure 1: The LG focused crawler overview.

Obviously, the candidate URLs found by our crawling procedure are not necessarily LG URLs. We need a classification procedure to further classify them into relevant and irrelevant ones. In this work, a two-step PU learning classifier including a *URL-based pre-filter* and a *content-based classifier*, is designed to efficiently and accurately classify the candidates. Specifically, the pre-filter based on extracted URL features filters out irrelevant URLs before downloading html files to save significant resources. As for the remaining URLs, which are named as *pre-filtered URLs*, we download their html files and use the content-based classifier based on carefully selected html file features to further classify them more accurately. Besides, the employed PU learning algorithm can overcome the challenge of lacking labeled negative samples. The pages that are confirmed by the content-based classifier to be LG pages are named as *relevant URLs*.

Whenever relevant URLs are discovered, they can be used as known LG pages to start a new round of iteration. The iterative exploration can help us find more relevant URLs.

2.2.1 LG seed set. We collect a set of known LG pages from public LG portal pages such as PeeringDB [8], Traceroute.org [14], BGP4.as [2], and BGPLookingglass.com [3]. In total, we have 2,991 known LG URLs to serve as the seeds to start our search process. These pages are downloaded using the python library Requests [9]. Some URLs respond with HTTP error messages and 1,736 html files are downloaded successfully. Some files are found to have expired and do not provide looking glass services now, *e.g.*, *https://www.gtanet.ca/looking-glass/*. After a manual check, we obtain 1085 valid html files that are providing looking glass services now.

2.3 Crawling procedure design

Many previous works have noticed that web page authors usually intend to create links to pages on the topics related to their own pages [25]. Therefore, they usually use hyperlink-guided search. In this paper, we also implement the hyperlink-guided search by directly extracting hyperlinks from known LG pages as candidate URLs. However, it is not sufficient because there can be many LG pages that are not connected to any other LG pages, and they cannot be found by this method.

To deal with the challenge, we develop a similarity-guided search method. LG pages are all deployed to provide LG services, and it is natural that they are likely to be similar in some aspects. To locate more effective candidate URLs, the basic idea of our similarity-guided search is to find pages that are similar to known LG pages. Intuitively, we can define metrics to evaluate the similarity between any two pages and then crawl the Internet and evaluate every page found by the crawler. But we all know that crawling the Internet is not easy and it is what we want to avoid. In this work, we exploit well-developed search engines (such as Google and Bing) to accomplish the crawling. These search engines have deployed excellent infrastructure to regularly crawl the Internet and answer queries from users. We analyze all known LG pages to find the common features shared by them. These shared features are transformed into *search terms*, and these search terms are entered into search engines to get candidate LG URLs. In this way, we do not need to crawl the Internet by ourselves and the professional search infrastructure and search algorithms of search engines can help us get high-quality candidate URLs.

The key issue is how to find the common features shared by LG pages and transform them into efficient search terms.

For example, a query using "looking glass" as the search term returns many pages that are related to mirror instead of the looking glass services we are talking about. This search term is not efficient and its results would consume a lot of resources if they are sent to our verifier for further analysis. Considering a web page usually contains three components, *i.e.*, URL, Title, and Body, which have different characteristics, we utilize appropriate methods to analyze seed pages on each of these three components to mine common features and construct corresponding search terms.

2.3.1 Title-based search terms. In view that the page title is a short description of a web page and usually contains only a few words, the popular frequent itemset mining algorithm Apriori [17] can be used to extract frequent words or phrases from a list of LG titles as shared features.

One point we should notice is that some page titles include the names or AS numbers of the organizations that deploy the LG pages. Treating these names and numbers as the same keyword can give us more valuable information on the features shared by LG page titles. Specifically, we replace all names and AS numbers with two virtual words, namely *ORG* and *ASN*, before we extract frequent phrases. After extracting, if these two virtual words are found in the frequent phrases, we will replace them with the name and the number of every AS on the Internet to construct a series of search terms.

In the above process, it is challenging to determine which words in titles represent the names of organizations. Although CAIDA's AS to Organization Mapping Dataset [5] provides the organization name of each individual AS, different organizations may have different naming conventions when embedding their names in LG page titles, which can be full names, initials, or other abbreviations. Wrong matching and replacing will affect the accuracy of extracted shared features. To deal with the problem, we find that LG page titles are more likely to embed the second-level domain names of company websites to indicate the organization names. For example, an LG page[1] title "Milecom Looking Glass" does not contain the corresponding full organization name "LLC Milecom" but contain "milecom" (the second-level domain of the company website *http://www.milecom.ru*). Therefore, we collect the company website URL of each individual AS from PeeringDB and get 13K URLs in total. These URLs are parsed using the Tldextract python library [13] and we obtain a list of second-level domains. Then we can replace the word or phrase (in titles) that matches one second-level domain with *ORG*.

After replacing, we treat each title as a transaction and the title words as items and use the Apriori algorithm to mine frequent itemsets. Table 1 shows the frequent itemsets and their corresponding support when the support threshold is specified to be 0.12. We find that each of the frequent 1-itemsets and 2-itemsets is a proper subset of one frequent 3-itemsets, which means the search results from the frequent 1-itemsets and 2-itemsets are supersets of the search results from the frequent 3-itemsets. To balance the recall rate and accuracy rate, we choose the frequent 3-itemsets to construct search terms.

frequent 1-itemsets		frequent 2-itemsets		frequent 3-itemsets	
1-itemset	support	2-itemset	support	3-itemset	support
$\{looking\}$	0.805	$\{looking, glass\}$	0.804	$\{looking, glass, ORG\}$	0.343
$\{glass\}$	0.805	$\{ORG, looking\}$	0.344	$\{looking, glass, ASN\}$	0.194
$\{ORG\}$	0.403	$\{ORG, glass\}$	0.343		
$\{ASN\}$	0.220	$\{ASN, glass\}$	0.194		
		$\{ASN, looking\}$	0.194		

Table 1: The frequent itemsets with their corresponding support when the support threshold is set 0.12.

2.3.2 Body-based search terms. The body of LG pages usually contains richer information compared to their titles and URLs, which means analyzing bodies might be more helpful for us to construct efficient search terms. On the other hand, it is also more complicated than analyzing titles and URLs. Page bodies can contain many words, and some words appear frequently but they do not suggest any common features of LG pages. For example, the frequent words *network* and *help* in known LG bodies are also frequently used in webpages that are not related to any looking glass services[2]. Therefore, we choose to use the TF-IDF (Term Frequency–Inverse Document Frequency) [50] weighting model to analyze page bodies. The TF-IDF can identify words that frequently appear in known LG bodies but do not frequently appear in non-LG pages, thereby helping to construct more efficient search terms.

In order to reduce the computation complexity, we first extract informative texts from all valid seed html files and these texts are merged into a document, which is referred to as the *text-document*. We notice that most LG pages usually contain html control elements such as *input*, *select*, and *button* to enable users to conduct measurements using their looking glass services. Therefore, we use the lightweight python library *Beautiful Soup* [1] to extract the texts between the opening and closing tags inside these three elements from each individual LG page and construct the text-document.

Now we can use the TF-IDF model to analyze the text-document and retrieve frequent words that exclusively appear in the text-document. This model requires an IDF corpus to be used to exclude frequent words that appear in all kinds of documents. For this purpose, we collect 11,314 documents

[1] *https://lg.milecom.ru/*

[2] Such as *https://www.foxnews.com/* and *https://abcnews.go.com/*

from 20Newsgroups [38] as the IDF corpus. In the TF-IDF weighting model, each text word t in the text-document is assigned a weight w_t, which is calculated as follows,

$$\begin{aligned} w_t &= TF_t \times IDF_t \\ IDF_t &= \log \frac{1+N}{1+n_t} + 1. \end{aligned} \tag{1}$$

Here, TF_t (term frequency) means the frequency of t in the text-document, while IDF_t (inverse document frequency) stands for the inverse of the frequency of t in other documents, $N = 11,314$ in our case, and n_t is the number of documents collected from 20Newsgroups that include the word t.

Obviously, the words with high w_t frequently appear in the text-document but do not appear frequently across other documents, therefore their appearance is likely to suggest an LG page. In this way, we extract 49 words (such as *ip_address*, *router*, *traceroute*, etc.) with $w_t > 0.05$. Each individual word combined with the word *looking glass* becomes a search term.

2.3.3 URL-based search terms. In fact, search engines are running the algorithm to rank pages according to their relevance to the search term. Therefore, we can exploit search engines directly and input 2,991 known LG URLs as search terms to get candidate URLs. For example, if we use *https://www.sprint.net/tools/looking-glass* as the search term in Bing, the search results will include another LG URL *https://lookingglass.centurylink.com/*. As a result, each LG URL can serve as a search term.

2.4 Classification procedure design

In the previous subsection, we collect a lot of candidate URLs, which are likely but not necessarily providing looking glass services. In other words, some of them are relevant URLs and some are irrelevant URLs. In this subsection, we design a classifier to classify all candidates into relevant or not. The irrelevant URLs will be discarded to avoid unnecessary resource consumption in exploring them deeper.

2.4.1 Design considerations. There are mainly two considerations in designing the classifier.

Our selected classification algorithm must be able to deal with the problem of lacking labeled non-LG pages (negative samples). Classifying candidate URLs as either relevant or irrelevant is a binary classification problem. Traditional supervised classification algorithms, such as SVM [22] and decision tree [40], require both positive and negative labeled samples for training. However, in this work, we only have a small set of known LG pages (positive samples) and a large number of unlabeled candidate URLs, making the supervised classification algorithms infeasible. On the other hand, if we take unsupervised algorithms to solve our classification problem, the labeled information of known LG pages will be wasted so the classification results may be unsatisfactory. In this paper, we take a PU learning algorithm [27], which can train a classifier to distinguish between positive and negative instances in the unlabeled set given a small set of positive samples and a set of unlabeled samples.

To classify candidate URLs efficiently and correctly, we need to carefully select features. To accurately classify candidate URLs, we should exploit all available information about them, e.g., their html files. However, the candidates obtained from the crawling procedure are only URLs without downloaded html files. Downloading the html files that correspond to a large number of candidate URLs is time-consuming and bandwidth-intensive, making the classification process inefficient. We notice the URLs of LG pages are usually different from that of non-LG pages. Therefore, we would like to try to pre-filter out some irrelevant pages based on only URL features. The pages that are pre-filtered out do not need to be downloaded. We just download the remaining pages to classify them based on both URL and carefully selected html content features. Specifically, we develop a two-step classifier, including a *URL-based pre-filter* and a *content-based classifier*, which can have good efficiency and accuracy.

2.4.2 Feature extraction. PU learning models only work with fixed-length numeric inputs, while both URLs and html files consist of strings of various lengths. Therefore, we need to conduct feature extraction which builds appropriate feature vectors from the raw data of URLs or html files. The feature vectors should be informative and non-redundant to facilitate subsequent training and classifying steps.

Extracting feature from URLs. The features extracted from URLs are used to construct our pre-filter. To convert a collection of URLs into fixed-length feature vectors, we use a typical method of text feature extraction, namely the bag-of-words model. It first creates an ordered vocabulary including all unique words in the URLs of the collection, and then each unique word gets an integer index. For every URL in the collection, the bag-of-words model constructs its feature vector U, and U_i is set to be the times the word whose index is equal to i appears in the URL.

Extracting feature from html files. Intuitively, html content may contain richer information compared with URLs. Therefore, the candidate URLs that pass through the pre-filter will be further checked by a content-based classifier, which is constructed based on features of URLs and html content. Of course, not all texts within html files are informative and uninformative texts may negatively affect the classification results of the PU-Bagging algorithm. As introduced in §2.3, page titles and texts inside *input*, *select*, and *button* elements in page bodies are valuable to distinguish between relevant and irrelevant URLs. Additionally, values of *id* attribute, *name* attribute, and *value* attribute inside the above three elements are also informative. They are not considered in constructing search terms in §2.3 because search engines usually ignore all attribute values when they answer queries from users. We combine all carefully selected informative texts extracted from the URL and the html file of a page, and use the bag-of-words model to transform the texts into a numeric feature vector to serve as a representation of the page.

2.4.3 PU-Bagging model training. Both the URL-based pre-filter and content-based classifier aim to classify candidate URLs as relevant (positive class) or not (negative class) using only a small set of known LG pages (positive labeled samples). In this work, we choose the PU-Bagging algorithm [45], which is a typical PU learning approach, to complete the pre-filter and classifier.

PU-Bagging algorithm employs the bootstrap aggregation technique (named as *bagging*) to obtain an aggregated classifier from positive and unlabeled samples. To bypass the issue of lacking labeled negative instances, it repeatedly draws random bootstrap samples from unlabeled samples as negatives, and trains a supervised classifier through the drawn negative and known positive samples. Then multiple classifiers are aggregated into a classifier, which can reduce the variance induced by the randomness in selecting "negative" samples. Mordelet *et. al.* [45] reported that the PU-Bagging algorithm can achieve outstanding performance and fast running speed.

Algorithm 1 PU-Bagging

```
Require: $\mathbb{P}$, $\mathbb{U}$, $N$ = number of base classifier, $K$ = size of
    bootstrap samples, $T$ = threshold
Ensure: a classification function $r \longrightarrow \{0, 1\}$
 1: Initialize $\forall x \in \mathbb{U}, n(x) \longleftarrow 0, f(x) \longleftarrow 0$
 2: for each $n = 1$ to $N$ do
 3:     Draw a bootstrap sample set $\mathbb{U}_n$ of size $K$ in $\mathbb{U}$
 4:     Train a SVM classifier $f_n$ using samples on $\mathbb{P}$ and $\mathbb{U}_n$
 5:     for each $x \in \mathbb{U} \setminus \mathbb{U}_n$ do
 6:         $f(x) \longleftarrow f(x) + f_n(x)$
 7:         $n(x) \longleftarrow n(x) + 1$
 8:     end for
 9: end for
10: $s(x) = f(x)/n(x)$ for $x \in \mathbb{U}$
11: Return
        $r(x) = \begin{cases} 0 & s(x) < T \\ 1 & s(x) >= T \end{cases}$
```

Here, we take the URL-based pre-filter as an example to introduce how the PU-Bagging algorithm works. Let $\mathbb{P}$ denotes the positive labeled samples set which consists of all the known LG URLs. The unlabeled samples set $\mathbb{U}$ consists of the candidate URLs. Each sample in the two sets is represented by a feature vector. The PU-Bagging algorithm takes the set $\mathbb{P}$ and $\mathbb{U}$ as inputs, repeats N rounds to train N binary base classifiers, and averages their predictions as classification results of the aggregated classifier. The details of the algorithm are shown in Algorithm 1.

Similarly, our content-based classifier is also trained using the PU-Bagging algorithm. In the training, the algorithm takes as inputs the set of known LG pages ($\mathbb{P}$) and the set of remaining candidate pages ($\mathbb{U}_r$) which are classified to be relevant by the URL-based pre-filter.

3 EXPERIMENTAL AND EVALUATION RESULTS

We implement the above focused crawler and present experimental and evaluation results in this section.

3.1 Effectiveness of the similarity-guided search

Taking the first round of iteration (whose input is the LG seed set) as an example, we conduct the hyperlink-guided search to extract hyperlinks from 1,736 successfully downloaded seed html files and collect 4,436 unique candidate URLs. In the meanwhile, we conduct the similarity-guided search to analyze the URL, title, and body component of LG seed pages and obtain 100,987 search terms. Specifically, 97,947 of them are constructed by replacing *ORG* and *ASN* in the two frequent 3-itemsets mined from titles, 2,991 of them are constructed by taking each individual LG seed URL as a search item, and 49 search terms are constructed by analyzing LG page bodies. Each search term is entered into the Bing search engine, and from each search we collect at most top 10,500 returned search results as candidate LG URLs.

To evaluate the effectiveness of the similarity-guided search, we define four metrics: the number of relevant URLs obtained from crawled candidate URLs (after the processes in §2.4), the relevant URL concentration rate (the fraction of crawled candidate URLs that are classified as relevant), the number of obscure automatable LG VPs finally obtained from crawled candidate URLs (after the processes in §4.1), the automatable VP concentration rate (which equals the number of obtained obscure automatable LG VPs divided by the number of candidate URLs). These metrics are good measures of the gains from and the efficiency of our similarity-guided search.

Table 2 shows an overview of the results of the experiments. It is observed the similarity-guided search helps us obtain 4,111 relevant URLs and 608 obscure automatable LG VPs in total, which is about 24 (13) times more than relevant URLs (obscure automatable LG VPs) from the hyperlink-guided search. The results indicate that our similarity-guided search, which benefits from carefully constructed search terms and

	Search Terms	Candidate URLs	Relevant URLs	Relevant URL Concentration	Obscure VPs	VP Concentration
Hyperlink guided	—	4,436	147	3.31%	48	1.08%
Similarity guided	97,947 (Title)	461,799	2,511	0.54%	423	0.09%
	2,991 (URL)	433,865	1,901	0.44%	324	0.07%
	49 (Body)	19,793	1,114	5.63%	470	2.37%
	100,987 (ALL)	877,021	4,111	0.47%	608	0.07%
ALL	100,987	919,893	4,226	0.48%	630	0.07%

Table 2: An overview of experimental results provided by running the hyperlink-guided search and the similarity-guided search.

the well-developed search engine, can effectively find many relevant pages and obscure LG VPs that cannot be found by the widely used hyperlink-guided search. Besides, we find that the search terms constructed from page bodies can produce search results with larger concentration rates than the search terms from URLs and titles. As shown in table 2, the relevant URL concentration rate and the VP concentration rate of the body-based search terms are 5.63% and 2.37% respectively, which are much higher than those of the URL-based search terms (0.44% and 0.07%) and the title-based search terms (0.54% and 0.09%). Of course, it does not mean that URL-based and title-based search terms have no value, they can also help find many relevant pages and obscure LG VPs that cannot be found by the body-based search terms.

Please note the hyperlink-guided search and similarity-guided search will be conducted every time we find new relevant LG pages. Such iteration is effective for obtaining more relevant URLs and obscure VPs. Besides the results of the first iteration described above, in later three iterations, we additionally collect 2,047,333 candidate URLs and obtain 46,551 relevant URLs and 280 obscure automatable VPs.

3.2 Effectiveness of the two-step classifier

To evaluate the performance of the URL-based pre-filter and content-based classifier, we use the area under the receiver operating curve (AUC), the true positive rate (TPR), and the false positive rate (FPR) as evaluation metrics. The higher the AUC, the better the performance of the classifier at distinguishing between the positive and negative classes.

3.2.1 Evaluation on the URL-based pre-filter. We randomly split the whole dataset from the first round of iteration which has 2,991 LG seed URLs (positive URLs) and 919,893 unlabeled candidate URLs into three subsets: training dataset (98%), validation dataset (1%) and testing dataset (1%). Unlabeled URLs in the validation and testing dataset are manually labeled, thus the two datasets are fully labeled, which will be used for tuning hyperparameters and evaluating generalization ability respectively. Table 3 shows the statistics of these datasets. A URL is said to be "positive" if it provides looking glass services and otherwise it is said to be a "negative URL".

	Whole dataset	Training dataset		Validation dataset	Testing dataset
Positive URLs	2,991	2,931	Positive URLs	108	102
Unlabeled URLs	919,893	901,495	Negative URLs	9,121	9,127

Table 3: The number of different kinds of URLs for each dataset.

Hyperparameter tunning. There are some hyperparameters, *i.e.*, the number of base classifier N, the size of bootstrap samples K, and the classification threshold T, that need to be carefully chosen for training a good pre-filter. We train the URL-based pre-filter under different hyperparameters using the training set and get a series of trained pre-filters. According to their AUC performance on the validation set, we choose N to be 100, K equal to the number of positives in the training set (*i.e.* 2931) as optimal hyperparameters, which can help train a pre-filter with a high AUC of 0.9657.

In addition, the values of T will have a great impact on classification results. The results can be measured by TPR and FPR. A high TPR means most candidate URLs that are really providing looking glass services are classified as relevant. A low FPR means few candidate URLs that are not providing looking glass services are classified as relevant. For the pre-filter, to some extent, a high TPR is more important than a low FPR, because a low TPR rate means a lot of LG sites are filtered out incorrectly, while a low FPR just results in more resource overhead consumed by the later content-based classifier. Figure 2(a) plots FPR and TPR of the trained pre-filter under different T on the validation dataset when $K = 2931$ and $N = 100$. The trained pre-filter with $T = 0.2072$ can achieve a high TPR of 99.07% at FPR of 15.54%, which is more preferred for us. Therefore, we choose T to be 0.2072.

Classification results. To evaluate the generalization ability of the trained pre-filter with the optimal hyperparameters, we run it on the labeled testing dataset, which contains 102 positives and 9,127 negatives. The running result shows that our pre-filter produces a high TPR of 96.08% at FPR of 15.36%, which is close to its performance on the validation dataset. We can see that, in the testing dataset, the pre-filter filters out 7,725 negative URLs, which helps us save the resources of downloading their html files. Meanwhile, it only filters out 4 positive URLs which have a relatively small impact on our final results.

The pre-filter performs well and we now use it to classify all 919,893 candidate URLs. 789,967 candidate URLs are classified as irrelevant and they are filtered out immediately. The remaining 129,926 candidate URLs are classified as relevant by the pre-filter, and we name them as *pre-filtered URLs*. These pre-filtered URLs will be further classified by using the content-based classifier. *Our designed pre-filter can reduce the number of candidate URLs to be downloaded by about 85% with almost no loss of LG URLs, significantly improving the efficiency of classification.*

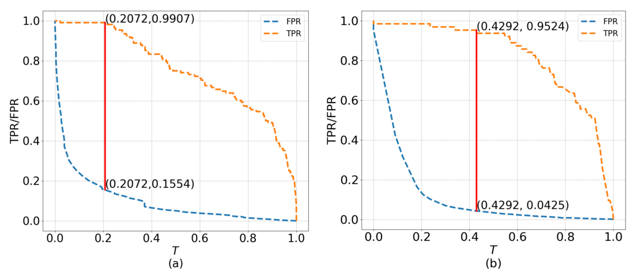

Figure 2: (a) The distribution of TPR and FPR of the trained pre-filter under different T. (b) The distribution of TPR and FPR of the trained classifier under different T.

3.2.2 *Evaluation on the content-based classifier.*

We download html files of the 129,926 pre-filtered URLs using the lightweight python library *Requests* [9]. Some URLs respond with error messages and we have to discard them. 77,113 URLs respond successfully and we get their html files. As introduced in §2.2.1, there are 1,085 valid seed html files that are providing looking glass services now. These successfully downloaded unlabeled html files and seed html files are used as the whole dataset to train the content-based classifier. Since the whole dataset here is smaller than that of the pre-filter, we need to increase the ratios of the validation and test dataset to the whole dataset to obtain sufficient evaluation samples. As a result, we randomly split the whole dataset into three subsets: training dataset (94%), validation dataset (3%), and testing dataset (3%). Unlabeled pages in the validation and testing dataset are manually labeled. The statistics of these datasets are shown in Table 4.

	Whole dataset	Training data		Validation data	Testing data
Positive pages	1,085	1020	Positive pages	63	77
Unlabeled pages	77,113	724,86	Negative pages	2,283	2,269

Table 4: The number of different kinds of pages for each dataset.

Hyperparameter tunning. Similar to the above process of determining hyperparameters for the pre-filter, we choose N to be 100 and K to be 1020 as optimal hyperparameters, which can help train a classifier with a high AUC value of 0.97. We further determine T. Figure 2(b) plots FPR and TPR of the trained classifier under different T when $K = 1020$ and $N = 100$. For the content-based classifier, we would like to have both high TPR and low FPR, which means most truly positive pages will be classified as relevant, most truly negative pages will be filtered out and the probability for the obtained relevant pages to be LG pages is high. It can be seen that the trained classifier with $T = 0.4292$ can achieve a high TPR of 95.24% and a low FPR of 4.25% on the validation set. So we choose T to be 0.4292.

Classification results. To evaluate the generalization ability of the classifier, we run it on the labeled testing dataset, which consists of 77 positive pages and 2,269 negative pages. It shows our classifier has a high TPR (96.10%) and a low FPR (4.10%). In other words, in the testing dataset, the classifier filters out 2,176 negative pages, which can help us save the resources of trying to retrieve VPs from them. Meanwhile, it only drops 3 positive pages.

Now we use the content-based classifier to classify the 77,113 pre-filtered URLs. 4,226 URLs are classified as relevant by the classifier. We name them as *relevant URLs* and will further try to retrieve automatable VPs from these relevant pages for practical applications.

It worths noting that all relevant URLs will serve as new LG seed pages and we will conduct the crawling procedure based on these new seeds. The new search process will give us new candidate URLs and they will be further classified by our pre-filter and classifier. This process repeats iteratively, and the 50,777 relevant URLs from all iterations will be analyzed to facilitate practical applications later.

4 PRACTICAL APPLICATIONS

Through the focused crawler, we obtain relevant pages which have high probabilities to be LG pages. These relevant pages have been very useful for measurement researches. For example, it becomes feasible for a researcher to retrieve a list of available VPs by manually analyzing every relevant page, and manually use these VPs to conduct measurement commands. As for researchers who require large-scale or periodic measurements, it will be more practical if we automate the use of these VPs. In this section, we develop a tool to retrieve automatable LG VPs and show practical values of the automatable obscure LG VPs in facilitating measurement researches. The tool and the retrieved automatable LG VPs will be publicly accessible later.

4.1 Retrieving automatable LG VPs

LG sites usually have different input interfaces to collect and parse measurement requests. The lack of input interface standardization hinders the automatic use of LG VPs. Therefore, we need to develop a tool to automatically retrieve input interface information about VPs and check whether these LG

VPs are automatable by analyzing responses to measurement requests that are automatically issued by us.

4.1.1 Retrieving information about VPs. We design a template matching and a keyword matching method to retrieve the web-based input interface information about VPs by deriving more templates and matching relevant pages with the templates or keywords.

Matching with templates. Many deployments of LG services are based on several popular open-source projects [24, 29]. These open-source projects have different input interface specifications. In this work, we derive eight input interface templates by analyzing html files created by each project that is mentioned in papers [29] and [24]. Each template records the name, type, and purpose of every input field within the *form* element. We say that a relevant page matches a template if each pair of input field name and type within its *form* element is the same as the template, and the page is referred to as a *template-matching page.* In total, 1,302 relevant pages are found to be template-matching pages.

Matching with keywords. As for the remaining relevant pages, we find some of them have input fields that are intended for the same purpose as the input fields in the templates. However, the names of these input fields do not match the templates. We search for relevant pages that contain LG-specific keywords (*ping, trace, or bgp*) in their *form* elements, and refer to these pages as *keyword-matching* pages. Overall, 322 relevant pages are found to be keyword-matching pages.

For the template-matching and keyword-matching pages, we automatically summarize the information of each VP (i.e. all commands it supports and the required input fields to run each command and how to fill them) into a single file, named as the *VP meta-file.* Overall, we retrieve 3,848 VPs and they enable 17,753 commands in total. As a comparison, we also use the above methods to analyze the 1,085 seed pages that are in operation. 791 seed pages are found to be template-matching or keyword-matching pages. These pages provide 4,432 VPs which support 23,024 commands.

4.1.2 Automatically issuing measurement requests. With the VP meta-file, each measurement request can be translated to the action of filling in the input fields of the corresponding form element with specific values. Mechanize library is used to programmatically fill in the required input fields of the form and send the form to the corresponding web server. Once the server receives the form data, it will respond to the measurement request and return measurement results.

4.1.3 Analyzing responses to determine automatable VPs. If one VP can successfully respond to our measurement requests, we refer it to as an *automatable VP.* Specifically, we automatically issue ping measurement requests to ask the VP to ping a controlled machine, which runs *tcpdump* to capture all incoming ICMP packets. If the machine sees ICMP ping packets arriving, it means the VP is automatable. In the meanwhile, the IP address of the VP can be extracted from the ICMP packets. We can then learn the location of the VP from its IP using IP2AS and geolocation databases, which can facilitate researchers to select VPs when they need to conduct measurements from a specific country and AS.

Using the above methods, we successfully determine 1,446 automatable VPs from the seed LG pages, and 1,296 automatable VPs from the relevant pages. Some VPs from different pages refer to the same server/router (IP address), which means they are the same vantage point. After deduplication, we find that 910 automatable VPs from the relevant pages are not included in any seed page, which are referred to as *obscure automatable VPs.* Automatable VPs that are included in the seed pages are referred to as *known automatable VPs.*

4.1.4 Discussion. A relevant page is confirmed to be providing LG services if our tool can retrieve automatable VPs from it successfully. But it does not suggest that the page is not providing LG service even if the tool cannot retrieve automatable VPs from it. One reason is the diversity of LG pages makes it difficult to retrieve input interface information about all LG pages. Our tool can only deal with a part of LG pages. Another reason is that some LG pages restrict automated access. To obey their requirements, the way our tool issues measurement requests does not accomplish the reCAPTCHA verification, thus LG VPs from LG pages that include the reCAPTCHA cannot be retrieved by our tool.

4.2 Analysis and applications of the automatable VPs

Researchers have noticed the limited geographic and network coverage of available VPs hinders our comprehensive understanding of the Internet [31]. It would be valuable if our discovered obscure automatable VPs can support diverse measurement commands and bring geographic and network coverage improvements. We compare the 910 obscure automatable with known VPs in terms of geographic and network coverage to show their practical values. Moreover, we introduce a case study to demonstrate their potential values in improving Internet topology completeness.

4.2.1 Diverse measurement capabilities. Each automatable LG VP may allow the execution of diverse measurement commands. We plot the number of obscure and known automatable VPs that support each command in Figure 3. It can be seen that our work can significantly increase the number of automatable VPs that support each command, which is beneficial to both data plane and control plane research.

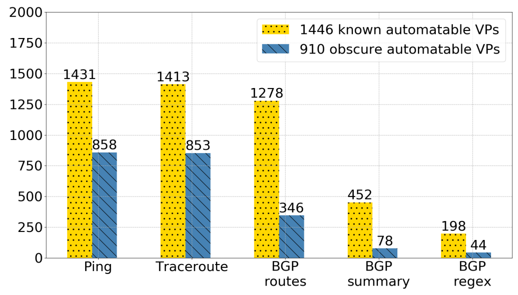

Figure 3: The number of automatable VPs that support each command.

4.2.2 Coverage improvements. In the following, we will analyze geographic and network coverage improvements that can be brought by our discovered obscure automatable VPs.

- Geographic coverage improvements.

Although there are some public IP geolocation databases, it is well known that obtaining accurate IP to geolocation mappings is still a challenge. We find the country and city level locations of VPs from pages that match the template of Telephone LG project [11] have been explicitly given in html files. We try to use this reliable geographic information. Through regular expressions matching, we successfully retrieve the country-level locations about 390 VPs and city-level locations about 316 VPs. For the remaining VPs, we use a paid IP2location DB9 database to help map them into country-level and city-level locations. By comparing the DB9 database with the reliable geographic information, we find the country-level accuracy of the DB9 database can achieve 97% when we apply the database to the 390 VPs with known country-level locations, which is acceptable.

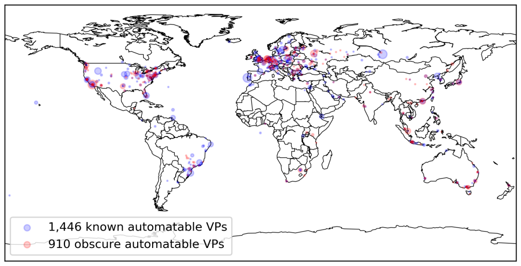

Figure 4: The geographic coverage of 1,446 known automatable VPs (blue points) and 910 obscure automatable VPs (red points).

Figure 4 plots the geographic coverage of 1,446 known automatable VPs (blue points) and 910 obscure automatable VPs (red points). The size of points represents the number of VPs in the corresponding geographic area. 1,446 known automatable VPs are distributed in 386 cities of 75 countries, and 910 obscure automatable VPs cover 282 cities in 55 countries. *The obscure VPs enable researchers to execute measurement commands from 8 new countries and 160 new cities where no known LG VPs have been found before.* In particular, the 8 countries are mainly distributed in East Africa and South Asia, whose network connectivities and performance have attracted a lot of attention of researchers recently.

- Network coverage improvements.

Inferring router ownership is also difficult. We notice some LG pages tend to explicitly give ASNs of their VPs in html files, which we think is a reliable VP2AS mapping source. Through regular expressions matching, we obtain the ASN information about 526 VPs. Then we use the bdrmapIT [42] tool to construct an IP2AS mapping dataset based on the public dataset of traceroutes from CAIDA's Archipelago (Ark) [12] in December 2019. The dataset can help obtain the ASNs of 323 VPs, which are not available from their html files. For the remaining VPs, we query the Routeviews Prefix to AS mappings dataset [10].

Our analysis results show that *the obscure automatable VPs enable researchers to execute measurement commands from 270 exclusive ASes where no known LG VPs have been found before.* The ASes in the Internet hierarchy can be divided into three tiers, i.e., tier-1, tier-2, and stub tier, according to the data collected from wiki [15]. We categorize the ASes of VPs and plot the numbers of ASes in each tier in Figure 5(a). It is observed that the 910 obscure VPs are more concentrated in the stub tier compared to the known VPs. Especially, 269 of the 270 exclusive ASes are all in the stub tier. The previous studies [30, 48] suggested the incompleteness of the AS-level topology can be improved by placing VPs on the edge of the Internet. Therefore, the VPs found by us are valuable for improving the completeness of the Internet topology.

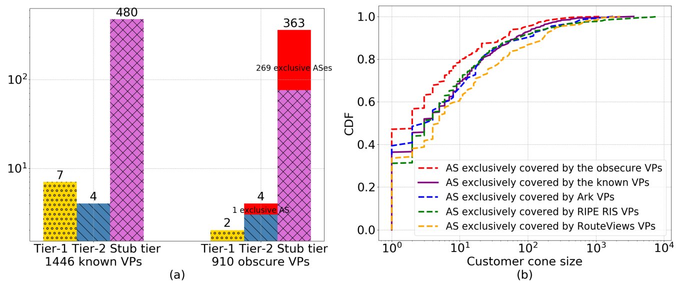

Figure 5: (a) Categorizing the ASes of VPs according to network tier; (b) The distribution of customer cone sizes of ASes covered by different VP datasets.

Besides web-based LG sites, there are other types of platforms providing VPs to collect measurement data. Some VPs just provide measurement results and do not allow users to run their own commands, and some VPs only allow privileged users. Although these platforms are different from LG sites, in order to comprehensively evaluate the value of our findings, we still compare our findings with the VPs provided on other platforms, such as RIPE RIS, Ark and RouteViews. Table 5 shows the number of ASes and exclusive ASes covered by the VPs of each platform. We find that the obscure automatable LG VPs exclusively cover 262 ASes, *i.e.*, there are no VPs deployed by all other platforms in these ASes.

	Ark VPs	RIPE RIS VPs	RouteViews VPs	Known LG VPs	Obscure LG VPs
ASes	115	520	270	491	369
Exclusive ASes	67	354	135	405	262

Table 5: The number of ASes and exclusive ASes covered by VPs of each platform.

The customer cone size [41] of one AS is also an indicator of its location in the hierarchical structure of the Internet. We collect the customer cone size of each AS from CAIDA's AS rank project [4], and plot the distribution of customer cone size of ASes exclusively covered by each dataset in Figure 5(b). It can be seen that about 47% of the 262 ASes exclusively covered by the obscure automatable VPs have a customer cone size of 1, and 80% have a customer cone size smaller than 10. Compared to the ASes exclusively covered by other platforms, our exclusive ASes tend to be at the edge of the Internet, which will bring different views of the Internet.

4.2.3 A case study in improving topology completeness. In the following, we introduce a simple case study in which we successfully collect AS-level routing paths from 8 obscure VPs (0.9% of the 910 VPs) that support both commands of *show ip bgp summary* and *show bgp neighbor ip advertised (or received) routes* to help improve topology completeness.

- Collecting AS paths from VPs.

AS paths returned by the *show bgp neighbor ip advertised (or received) routes* command can help construct the AS-level Internet topology. We note that some popular Looking Glass projects, such as HSDN [7] and Cougar [6], present input interfaces of *show bgp neighbor ip routes* command (if any) in the response page for a *show bgp summary* command.

Using the method of automatically issuing measurement requests developed in §4.1, we can automatically conduct the process of collecting AS paths from VPs. Specifically, with the VP meta-file, we programmatically fill in the required input fields to send the *show ip bgp summary* measurement request to every VP using Mechanize library. Each generated webpage will give the status of every BGP connection with the VP and the ASN and IP address of its neighboring BGP router for each BGP connection. Then, by visiting the hyperlink of each neighbor IP, we can ask the VP to run *show bgp neighbor ip* command to collect detailed information about the neighbor IP, including a hyperlink for showing its advertised (or received) routes. By further visiting the hyperlinks, we send requests to the VP to run the command to collect advertised (or received) BGP routes, where AS paths can be extracted using the regular expressions matching method. The script used for automatically collecting AS paths from VPs will be also publicly accessible.

- Improving AS-level topology completeness.

Using the above method, we successfully extract AS path information from 14 known VPs and 8 obscure VPs. As a comparison, we also construct AS topologies based on BGP data collected from two popular BGP collector projects: RIPE RIS and RouteViews. They deploy route collectors peering with commercial ISP networks via BGP sessions and collect BGP updates and RIB dumps from VPs in these ASes. We download BGP routing table snapshots observed from all RouteViews and RIS route collectors in December 2020.

		Known LG VPs	Obscure LG VPs	RIPE RIS	RouteViews	ALL
ASes	Observed	44,955	44,355	44,952	45,339	45,635
	Exclusive	247	10	12	271	-
AS links	Observed	100,356	76,907	154,828	204,889	253,719
	Exclusive	8,318	1,428	37,383	85,450	-

Table 6: The number of observed and exclusive ASes, AS links extracted from each dataset.

Table 6 shows the number of observed and exclusive ASes and AS links that are extracted from each dataset. We find using the 8 obscure VPs can exclusively observe 10 ASes and 1,428 AS links, which cannot be discovered by all other RIPE RIS VPs, RouteViews VPs, and 14 known LG VPs.

Discussion. Other discovered obscure VPs are also useful in improving the completeness of AS-level topology. In fact, the number of obscure automatable VPs that support *traceroute* or *show ip bgp routes* command is much more than that used in the case study (see Figure 3). For VPs that support *traceroute*, we can automatically ask them to traceroute to a list of targeted IP addresses and get a lot of paths. For the VPs that support *show ip bgp routes*, we can ask them to return BGP routes for a list of user-specified network prefixes. The problem to be solved is how to choose the target IPs for *traceroute* requests and how to specify the prefixes for *show ip bgp routes* requests. Due to the request rate limits of some VPs, the target IPs and prefixes should be carefully selected to achieve good performance in improving the topology completeness. We leave it as future work and do not use these VPs in this case study.

5 ETHICAL CONSIDERATIONS

Ethical issues have been considered in this work. The LG sites publicly accessible on the web are deployed to provide windows to understand the Internet and troubleshoot Internet connectivity and performance issues, therefore, discovering and using them to facilitate Internet measurement researches are reasonable applications. Besides, our work does not involve any operations that violate the requirements of LG sites. For example, when we retrieve automatable LG VPs, we did not try to bypass the verification code for LG pages that use the reCAPTCHA to restrict automated access. Another important ethical consideration is to avoid excessive overhead on LG sites when we use them. To this end, we limit the query rate to visit each LG site once every half an hour when we determine automatable VPs and use LG sites to collect AS-level paths in the case study. It worths noting that the system developed to automate the use of LG VPs in §4.1 is not for accelerating measurements, but only for providing convenience for periodic measurements. In addition, we also limit the number of search terms entered into the Bing search engine in a given time period when we conduct similarity-guided searching, thereby preventing abuse.

6 RELATED WORK

Researchers have noticed that LG VPs are valuable for measurement researches [20, 29, 37, 46, 47]. For example, in 2013, Khan et.al [37] find using VPs from 304 well-known LG sites can help discover 11K new AS links and 686 new ASes. However, existing works only use well-known LG sites. In this paper, we aim to discover the LG sites that are not published on well-known LG portal pages.

Focused crawlers are proposed to discover topic-specific pages on the web. Whenever a topic is given, a specific focused crawler needs to be designed with considerations of the special characteristics of pages on the topic [19, 28, 49, 52]. For example, to discover medicinal plant pages, Pawar et.al [49] argue that pages on the medicinal plant topic are usually well connected and so they take hyperlinks in known medicinal plant pages as candidate URLs to crawl more pages on this topic. As far as we know, there is no previous work on designing focused crawlers to discover LG sites on the web.

Researchers find only taking hyperlinks of known topic-relevant pages as candidate URLs can miss many topic-relevant pages [16, 23, 26, 28, 33]. There can be some relevant pages that are indirectly connected to known topic-relevant pages via multiple hops of irrelevant pages. So they propose to explore a tunneling technique that allows crawlers to extract hyperlinks of irrelevant pages to discover more relevant pages. Designing a focused crawler based on the tunneling technique is very sophisticated. Furthermore, in our scenario, many LG sites are isolated and they are not connected to known LG sites either directly or indirectly. The tunneling technique cannot help us discover these LG sites.

Several previous works notice well-developed search engines are valuable for designing a focused crawler to efficiently find more malicious web pages [34], academic slides [39]. Due to the different characteristics or different types of web pages, their methods of constructing search terms are not suitable for our problem. In this work, we design a similarity-guided search that exploits search engines and carefully mines search terms to find more LG pages.

The performance of a classifier depends on classification algorithms and selected features. In scenarios with only a small number of positive samples and a large number of unlabeled samples, PU learning algorithms have been used to avoid labeling training samples and achieved good performance [54–56]. As for the selection of features, it needs to adapt to different scenarios [18, 36, 43, 44, 53]. For example, Jiyoung et.al [36] find link-based features, which represent relations or links among pages, can perform well in detecting spam pages due to their rich connection structure. Obviously, link-based features are not suitable for our scenario because of the poverty of links among LG sites. We carefully observe the properties of LG sites and design appropriate URL-based features and content-based features to classify pages.

The challenge of automatic use of LG VPs has drawn attention from researchers in Periscope [29]. Periscope introduces a matching method to unify disparate interfaces of well-known LG VPs into a standardized querying API. It only focuses on automating the use of VPs, while our work mainly aims to discover obscure LG sites on the web.

7 CONCLUSION

In this article, we design an efficient focused crawler to discover obscure LG sites on the web. Faced by the challenge that there exist poor hyperlink connections between LG sites, our crawler designs a similarity-guided search that exploits well-developed search engines and comprehensively mines the common features shared by known LG sites. Results show that the similarity-guided search can help discover more LG sites. To classify candidate URLs into relevant or not, we design a two-step PU learning classifier based on carefully selected LG features to provide good classification performance. The URL-based pre-filter can help us save significant resource consumption in downloading a large number of html files of irrelevant URLs. Moreover, compared to supervised learning based classifiers, our PU learning classifier can avoid the time-consuming work to manually label numerous samples. For the relevant pages obtained through the focused crawler, we develop a tool to retrieve automatable LG VPs to further facilitate practical applications.

In total, we discover 910 obscure automatable LG VPs which are not published on any well-known LG portal pages. Our work significantly increases the number of automatable VPs that are available to researchers, which is only 1,446 before our work (from all well-known LG sites). Besides, the 910 obscure automatable LG VPs enlarge the geographic and network coverage of available VPs and present potential values in improving topology completeness in a case study.

As time goes on, some newly created LG sites may appear on the web. In future work, we plan to periodically repeat the process to update the list of relevant pages and automatable LG VPs. The well-developed search engines allow us to specify a time period in each search request. In this way, we can ask the search engine to only return results that are newly indexed, which facilitates our incremental crawling. The list of discovered relevant pages and automatable LG VPs, and the source code of our focused crawler system and automation tool will be public accessible, which can be very valuable for future measurement researches.

REFERENCES

[1] [n.d.]. Beautiful Soup. Retrieved August, 2020 from https://pypi.org/project/beautifulsoup4/

[2] [n.d.]. BGP4.as. Retrieved April, 2020 from http://www.bgp4.as/looking-glasses

[3] [n.d.]. BGPlookingglass.com. Retrieved April, 2020 from http://www.bgplookingglass.com

[4] [n.d.]. CAIDA AS Rank. Retrieved October, 2020 from http://as-rank.caida.org/

[5] [n.d.]. The CAIDA UCSD AS to Organization Mapping Dataset. Retrieved April, 2020 from https://www.caida.org/data/as_organizations.xml

[6] [n.d.]. Cougar Looking Glass. Retrieved September, 2020 from https://github.com/Cougar/lg

[7] [n.d.]. HSDN Looking Glass. Retrieved September, 2020 from https://github.com/hsdn/lg

[8] [n.d.]. PeeringDB. Retrieved April, 2020 from http://www.peeringdb.com

[9] [n.d.]. Requests. Retrieved June, 2020 from https://pypi.org/project/requests/

[10] [n.d.]. Routeviews Prefix to AS mappings Dataset for IPv4 and IPv6. Retrieved September, 2020 from https://www.caida.org/data/routing/routeviews-prefix2as.xml

[11] [n.d.]. Telephone Looking Glass. Retrieved September, 2020 from https://github.com/telephone/LookingGlass

[12] [n.d.]. The CAIDA UCSD IPv4 Routed /24 Topology Dataset. Retrieved December, 2019 from https://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml

[13] [n.d.]. Tldextract. Retrieved June, 2020 from https://pypi.org/project/tldextract/

[14] [n.d.]. Traceroute.org. Retrieved April, 2020 from http://www.traceroute.org

[15] [n.d.]. Wiki. Retrieved September, 2020 from https://en.wikipedia.org/wiki/Tier_1_network

[16] Ahmed Abbasi, Tianjun Fu, Daniel Zeng, and Donald Adjeroh. 2013. Crawling credible online medical sentiments for social intelligence. In *2013 International Conference on Social Computing*. IEEE, 254–263.

[17] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*. 207–216.

[18] Hamidreza Alvari, Paulo Shakarian, and JE Kelly Snyder. 2017. Semi-supervised learning for detecting human trafficking. *Security Informatics* 6, 1 (2017), 1–14.

[19] Amalia Amalia, Dani Gunawan, Atras Najwan, and Fathia Meirina. 2016. Focused crawler for the acquisition of health articles. In *2016 International Conference on Data and Software Engineering (ICoDSE)*. IEEE, 1–6.

[20] Brice Augustin, Balachander Krishnamurthy, and Walter Willinger. 2009. IXPs: mapped?. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*. 336–349.

[21] Vaibhav Bajpai, Steffie Jacob Eravuchira, and Jürgen Schönwälder. 2015. Lessons learned from using the Ripe Atlas platform for measurement research. *ACM SIGCOMM Computer Communication Review* 45, 3 (2015), 35–42.

[22] Vanshita R Baweja, Rajesh Bhatia, and Manish Kumar. 2020. Support Vector Machine-Based Focused Crawler. In *Inventive Communication and Computational Technologies*. Springer, 673–686.

[23] Donna Bergmark, Carl Lagoze, and Alex Sbityakov. 2002. Focused crawls, tunneling, and digital libraries. In *International Conference on Theory and Practice of Digital Libraries*. Springer, 91–106.

[24] Luca Bruno, Mariano Graziano, Davide Balzarotti, and Aurélien Francillon. 2014. Through the looking-glass, and what eve found there. In *8th {USENIX} Workshop on Offensive Technologies ({WOOT} 14)*.

[25] Brian D Davison. 2000. Topical locality in the web. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. 272–279.

[26] Wei Dong, Hong Ni, Haojiang Deng, and Liheng Tuo. 2015. Gray Tunneling Based on Joint Link for Focused Crawling. In *3rd International Conference on Mechatronics, Robotics and Automation*. Atlantis Press, 859–862.

[27] Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 213–220.

[28] Mohamed MG Farag, Sunshin Lee, and Edward A Fox. 2018. Focused crawler for events. *International Journal on Digital Libraries* 19, 1 (2018), 3–19.

[29] Vasileios Giotsas, Amogh Dhamdhere, and Kimberly C Claffy. 2016. Periscope: Unifying looking glass querying. In *International Conference on Passive and Active Network Measurement*. Springer, 177–189.

[30] Enrico Gregori, Alessandro Improta, Luciano Lenzini, Lorenzo Rossi, and Luca Sani. 2012. On the incompleteness of the AS-level graph: a novel methodology for BGP route collector placement. In *Proceedings of the 2012 Conference on Internet Measurement Conference (IMC)*. 253–264.

[31] Enrico Gregori, Alessandro Improta, Luciano Lenzini, Lorenzo Rossi, and Luca Sani. 2014. A novel methodology to address the Internet AS-level data incompleteness. *IEEE/ACM Transactions on Networking* 23, 4, 1314–1327.

[32] Enrico Gregori, Luciano Lenzini, and Valerio Luconi. 2017. AS-Level Topology Discovery: Measurement strategies tailored for crowdsourcing systems. *Computer Communications* 112 (2017), 47–57.

[33] Miyoung Han, Pierre-Henri Wuillemin, and Pierre Senellart. 2018. Focused crawling through reinforcement learning. In *International Conference on Web Engineering*. Springer, 261–278.

[34] Luca Invernizzi, Paolo Milani Comparetti, Stefano Benvenuti, Christopher Kruegel, Marco Cova, and Giovanni Vigna. 2012. Evilseed: A

guided approach to finding malicious web pages. In *2012 IEEE symposium on Security and Privacy*. IEEE, 428–442.

[35] Zitong Jin, Xingang Shi, Yan Yang, Xia Yin, Zhiliang Wang, and Jianping Wu. 2020. TopoScope: Recover AS Relationships From Fragmentary Observations. In *Proceedings of the 2020 Conference on Internet Measurement Conference (IMC)*. 266–280.

[36] Joyce Jiyoung Whang, Yeonsung Jung, Seonggoo Kang, Dongho Yoo, and Inderjit S. Dhillon. 2020. Scalable Anti-TrustRank with Qualified Site-level Seeds for Link-based Web Spam Detection. In *Companion Proceedings of the Web Conference 2020*. 593–602.

[37] Akmal Khan, Taekyoung Kwon, Hyun-chul Kim, and Yanghee Choi. 2013. AS-level topology collection through looking glass servers. In *Proceedings of the 2013 Conference on Internet Measurement Conference (IMC)*. 235–242.

[38] Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*. Elsevier, 331–339.

[39] Jae-Gil Lee, Donghwan Bae, Sansung Kim, Jungeun Kim, and Mun Yong Yi. 2019. An effective approach to enhancing a focused crawler using Google. *The Journal of Supercomputing* (2019), 1–18.

[40] Jun Li, Kazutaka Furuse, and Kazunori Yamaguchi. 2005. Focused crawling by exploiting anchor text using decision tree. In *Special interest tracks and posters of the 14th international conference on World Wide Web (WWW)*. 1190–1191.

[41] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and KC Claffy. 2013. AS relationships, customer cones, and validation. In *Proceedings of the 2013 conference on Internet measurement conference (IMC)*. 243–256.

[42] Alexander Marder, Matthew Luckie, Amogh Dhamdhere, Bradley Huffaker, KC Claffy, and Jonathan M Smith. 2018. Pushing the boundaries with bdrmapit: Mapping router ownership at Internet scale. In *Proceedings of the 2018 conference on Internet Measurement Conference (IMC)*. 56–69.

[43] Srdjan Matic, Costas Iordanou, Georgios Smaragdakis, and Nikolaos Laoutaris. 2020. Identifying Sensitive URLs at Web-Scale. In *Proceedings of the 2020 Conference on Internet Measurement Conference (IMC)*. 619–633.

[44] Luke K McDowell, Aaron Fleming, and Zane Markel. 2014. Evaluating and extending latent methods for link-based classification. In *Workshop on Formal Methods Integration*. Springer, 227–256.

[45] Fantine Mordelet and J-P Vert. 2014. A bagging SVM to learn from positive and unlabeled examples. *Pattern Recognition Letters* 37 (2014), 201–209.

[46] Reza Motamedi, Bahador Yeganeh, Balakrishnan Chandrasekaran, Reza Rejaie, Bruce M Maggs, and Walter Willinger. 2019. On mapping the interconnections in Today's Internet. *IEEE/ACM Transactions on Networking* 27, 5 (2019), 2056–2070.

[47] George Nomikos, Vasileios Kotronis, Pavlos Sermpezis, Petros Gigis, Lefteris Manassakis, Christoph Dietzel, Stavros Konstantaras, Xenofontas Dimitropoulos, and Vasileios Giotsas. 2018. O Peer, Where Art Thou? Uncovering Remote Peering Interconnections at IXPs. In *Proceedings of the 2018 Conference on Internet Measurement Conference (IMC)*. 265–278.

[48] Ricardo Oliveira, Dan Pei, Walter Willinger, Beichuan Zhang, and Lixia Zhang. 2009. The (in) completeness of the observed internet AS-level structure. *IEEE/ACM Transactions on Networking* 18, 1 (2009), 109–122.

[49] Nisha N Pawar and K Rajeswari. 2016. Study of different focused web crawler to search domain specific information. *International Journal of Computer Applications, 2016,* 136, 11 (2016).

[50] Juan Ramos et al. 2003. Using TF-IDF to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, Vol. 242. New Jersey, USA, 133–142.

[51] Yuval Shavitt and Udi Weinsberg. 2009. Quantifying the importance of vantage points distribution in Internet topology measurements. In *IEEE INFOCOM 2009*. IEEE, 792–800.

[52] AK Singh and Navneet Goyal. 2017. Malcrawler: A crawler for seeking and crawling malicious websites. In *International Conference on Distributed Computing and Internet Technology*. Springer, 210–223.

[53] Harshal Tupsamudre, Ajeet Kumar Singh, and Sachin Lodha. 2019. Everything Is in the Name–A URL Based Approach for Phishing Detection. In *International Symposium on Cyber Security Cryptography and Machine Learning*. Springer, 231–248.

[54] Yingchao Wu, Qinghua Zheng, Yuda Gao, Bo Dong, Rongzhe Wei, Fa Zhang, and Huan He. 2019. TEDM-PU: A Tax Evasion Detection Method Based on Positive and Unlabeled Learning. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 1681–1686.

[55] Peng Yang, Xiaoli Li, Hon-Nian Chua, Chee-Keong Kwoh, and See-Kiong Ng. 2014. Ensemble positive unlabeled learning for disease gene identification. *PloS one* 9, 5 (2014), e97079.

[56] Ya-Lin Zhang, Longfei Li, Jun Zhou, Xiaolong Li, Yujiang Liu, Yuanchao Zhang, and Zhi-Hua Zhou. 2017. Poster: A PU learning based system for potential malicious url detection. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2599–2601.

A APPENDIX

To help researchers reproduce all the results presented in the paper, we make all artifacts of our paper publicly accessible at *https://github.com/zhuangshuying18/discover_obscure_LG*.

The artifacts include:

- Source Code: The source code to generate similarity-guided search terms and search for candidate URLs (crawling procedure), the source code to train the two-step classifier and obtain relevant URLs (classification procedure), and the source code to retrieve automatable LG VPs and automatically collect AS paths from the automatable VPs (practical applications) are all publicly available.
- Original Data: Reproducing the full experiments needs to take a very long time, especially the procedure to search for candidate URLs and the procedure to download html files of pre-filtered URLs. So we make our candidate URLs (from all iterations) and the downloaded html files publicly available. Then reproducers can save the time to do these two tasks and use our datasets directly. Furthermore, using our datasets, reproducers can avoid the influence of Internet dynamics.
- Trained Model: We publish our trained two-step classifier to eliminate the influence of the inherent randomness in the PU-bagging method on reproducing results.
- Analysis Results: We publish the analysis results used to fill in the tables and plot the figures in the paper.

Some of the artifacts are in fact tools. The two-step classifier is a classification tool for classifying URLs into LG-relevant or not. The source code to retrieve automatable VPs is an automation tool for automating the use of LG VPs. The script to collect AS paths from the automatable VPs is a measurement tool for collecting BGP entries.

We also publish the list of 1,446 known automatable VPs and 910 obscure automatable VPs, which is the final output of our research.