Gold Mining in a River of Internet Content Traffic

Zied Ben Houidi, Giuseppe Scavo, Samir Ghamri-Doudane, Alessandro Finamore, Stefano Traverso, Marco Mellia

Gold mining in a River of Internet Content Traffic*

Zied Ben Houidi[1], Giuseppe Scavo[1], Samir Ghamri-Doudane[1], Alessandro Finamore[2], Stefano Traverso[2], and Marco Mellia[2]

[1] Alcatel-Lucent Bell Labs, France
[2] Politecnico di Torino, Italy
{zied.ben_houidi,giuseppe.scavo1,samir.ghamri-doudane}@alcatel-lucent.com
{finamore,traverso,mellia}@tlc.polito.it

Abstract. With the advent of Over-The-Top content providers (OTTs), Internet Service Providers (ISPs) saw their portfolio of services shrink to the low margin role of data transporters. In order to counter this effect, some ISPs started to follow big OTTs like Facebook and Google in trying to turn their data into a valuable asset. In this paper, we explore the questions of what meaningful information can be extracted from network data, and what interesting insights it can provide. To this end, we tackle the first challenge of detecting "user-URLs", i.e., those links that were clicked by users as opposed to those objects automatically downloaded by browsers and applications. We devise algorithms to pinpoint such URLs, and validate them on manually collected ground truth traces. We then apply them on a three-day long traffic trace spanning more than 19,000 residential users that generated around 190 million HTTP transactions. We find that only 1.6% of these observed URLs were actually clicked by users. As a first application for our methods, we answer the question of which platforms participate most in promoting the Internet content. Surprisingly, we find that, despite its notoriety, only 11% of the user URL visits are coming from Google Search.

1 Introduction

The Internet ecosystem has been historically composed only by two actors: *carriers* (e.g., ISPs) offering telephony services and Internet connectivity, and *end-users*. Recently, a new class of actors progressively gained a prime role: *Over-The-Top content and application providers* (OTTs [1]), i.e., third parties that are involved in the cross-domain distribution of content. Nowadays, this class of providers offers to end-users the same services that used to be offered by ISPs, plus a plethora of novel services. This is the case for instance of voice, texting

* This work was supported by the European Commission under the FP7 IP Project "An Intelligent Measurement Plane for Future Network and Application Management" (mPlane).

and video services (e.g., Skype and Viber, WhatsApp and iMessage, or Netflix and YouTube).

In this context, ISPs are more and more assuming the low profitable role of "bits carriers". To counter this effect, ISPs have been trying to find new added-value services that go beyond the traditional network connectivity business, investing in platforms to offer content [2] or cloud services [3, 4].

However, one of the possible assets of an operator is the information about its customer-base. Some major ISPs such as AT&T and Verizon have recently investigated how to translate this asset into profit [5, 6], for instance by re-selling customers' data to advertisement agencies as already done by other big OTTs such as Facebook and Google.

Extending this concept further, ISPs are thinking of leveraging the information their networks carry. By looking at the traffic that naturally flows, they have access to a humongous river from which possibly gold mining valuable information. This information could be used for at least two use cases. First, for web analytics/marketing purposes. Second, it can serve as an input for novel recommendation systems based on network traffic observation. And, differently from most[3] OTTs, ISPs have a unique vantage point that offers a wide-angle perspective, where multiple services and customers' habits are possibly exposed. They can try to exploit this unique vantage point respecting end-users privacy and regulations, for example, avoiding targeting single users, but considering aggregate information only.

Putting ourselves in the same position of an ISP observing the traffic in its network, we first address the technical issues related to i) how to identify the "contents" that users access from the enormous set of "objects" the network carries, and ii) how to pinpoint "interesting content" that is worth recommending out of the previous set. Second, we contrast the information that ISPs could get against the one OTTs have. This is instrumental to quantify if the ISPs are really in a more rewarding vantage point compared to the OTT position.

In this work we focus on HTTP traffic, the standard protocol used to transport web objects[4]. We consider all URL requests that are passively observed on a link of an ISP network. Immediately, we realize that the very large fraction of these URLs consists of requests automatically generated by browsers to fetch objects that are part of a web page (images, CSS files, JavaScript code, etc.), or by applications to poll automatic services (e.g., software update, chat keep-alive, etc.). We refer to these as *browser-URLs*. We are instead interested in the small subset of URLs that were intentionally visited by users, which we call *user-URLs*. Lot of ingenuity has to be used to extract this second subset, and, to the best of our knowledge, little work has been conducted in the literature, with few methodologies that are either obsolete for nowadays scenarios [7, 8], or too complex for our needs [9],[10]. Our goal is to reach a good accuracy at

[3] Google for instance observes a lot of visits thanks to its widespread Google analytics tool.

[4] We ignore HTTPS for now. A discussion about how to deal with possible encryption is given in Sec. 6.

identifying *user-URLs* using only HTTP requests, without the need to correlate them with corresponding responses.

Next, we address the problem of identifying *interesting-URLs*, i.e., those user-URLs that are possibly worth recommending in an eventual content promoting scenario. For instance, a popular news is worth being suggested, while the home-page of a home banking portal is less interesting.

To develop and validate our methodologies we use a set of ground truth traces collected by manually visiting the top 1,000 most popular web sites rated by Alexa and 1,000 news promoted by Google News. Then, we show the goodness of our methodologies with a three-day long HTTP traffic trace passively collected from a PoP of an European ISP.

Our major contributions are:

• We present a first set of algorithms and filtering stages to extract user-URLs from passive HTTP traces. We tune and quantify the performance of the algorithms using the dataset for which we have the ground truth. Our results show that we successfully detect 96% of user-URLs with only 1% of false positive rate and 66% of precision. Besides, precision can be enhanced to 80% with a loss of only 6% of user-URLs. When applied to the ISP trace, we found that only 1.6% of the 190 million HTTP requests in the dataset corresponds to actual URLs intentionally visited by users.

• We propose two algorithms to detect which user-URLs should be promoted also as interesting-URLs. Both algorithms leverage the social network information that is typically present in popular and interesting web sites. We found that only 15% of user-URLs may be classified as interesting-URLs.

• We show how rich the perspective of an ISP can be. We observe users accessing videos hosted on 9 different platforms, and news coming from 80 different news portals. This indicates that the view most OTTs get is different from the wide-angle picture an ISP could get. We further demonstrate this view by providing early web analytics results on the referral shares of well known content promotion platforms. We show that, surprisingly, only 11% of all user-URL views happened after (thanks to) a Google Search, and, in the best case, no more than 13% of them came from Facebook, Twitter or Google News.

The rest of this paper is organized as follows. First, Sec. 2 describes the background and resumes the related work of this paper. Second, Sec. 3 describes the datasets we use. Sec. 4 describes our algorithms and presents their performance evaluation. Sec. 4 applies our filtering methods on real data. Finally, Sec. 6 presents the open problems of this work and concludes the paper.

2 Background and Related Work

The structure of a web page has considerably grown in complexity in the last decade. Today, it often consists of a complicated tangle of HTML, JavaScript, multimedia objects, CSS, XML files, etc. As depicted in Fig. 1, when attempting to view a web page, the browser downloads each of these objects using separate HTTP requests. As a consequence, a simple user's click on a URL

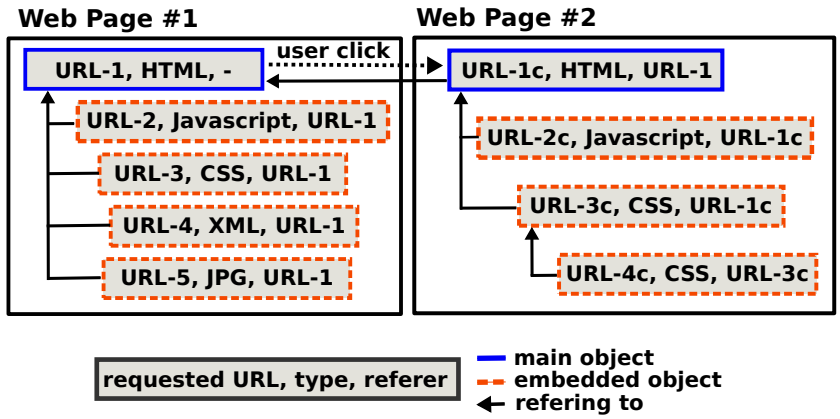

Fig. 1. Sequence of HTTP requests when accessing some standard web pages.

generates a cascade of several HTTP requests between the user terminal and an almost unpredictable number of servers and caches that store these objects. Besides, JavaScript, Ajax and Flash objects may generate extra HTTP traffic that was not intentionally requested by the user (e.g., the automatic fetching of a YouTube video). Furthermore, many non-web applications nowadays rely on HTTP to exchange their data (e.g., Dropbox and software updates). Therefore, even when a trace of HTTP transactions performed by a terminal is available, identifying which requests actually correspond to a human interaction is not a straightforward task.

To the best of our knowledge, only few methods have been proposed in the literature to detect URLs that were intentionally visited by users [8],[7],[9],[10].

The first method is time-based [8], it relies on the intuition that a cascade of browser-URLs follows the initial user-URL. Browser-URLs are tightly grouped in time after the user-URL request has been observed (see red dotted boxes in Fig. 1). As such, the first request after a given inactivity period can be considered as a new user-URL, while all the following HTTP requests happening before a time threshold are assumed to be browser-URLs. Unfortunately the estimation of the algorithm time thresholds is not a straightforward task, and many factors (e.g., latencies due to network conditions) may harm the performance of this method [9].

Another methodology, presented in [7], is type-based. It relies on the media information reported in the *Content-Type* field contained in the HTTP responses: the URLs associated to responses specifying text/HTML type are considered as main pages while the others are considered as browser-URLs. However, this approach has been shown to be unreliable [11].

To the best of our knowledge, the most recent methodologies to investigate the composition of modern web pages are StreamStructure [9] and ReSurf [10]. They both exploit the *Referer*[5] field to split HTTP transactions into multiple

[5] A field in the HTTP GET request that specifies from which previous page the URL has been referred.

streams, thus reducing the temporal overlaps between clicks on different URLs (e.g., when a user keeps several tabs open in her browser, for the same time window the technique can identify different streams). StreamStructure processes each detected stream separately. On each of them, it first applies a type-based filter (based on Content-Type) to keep only HTML/text pages. It then applies a time-based filter to detect main pages. To set a proper time threshold, the method leverages the Google Analytics beacon: once the Google Analytics beacon is downloaded, the page is considered as loaded, so that there is no need to set a static time threshold. Despite the improvements, StreamStructure still suffer from some limitations. First, it leverages Google Analytics information, which is present in only 40% of web pages (according to [9], [10]). Furthermore, it relies on the HTTP Content-Type field which has two drawbacks: (1) it has been shown to be often unreliable [11] and (2) this needs to monitor both HTTP requests and corresponding responses, which increases the complexity of the traffic extraction phase . We target a solution that overcomes these drawbacks, and for which only outgoing network traffic is needed to be monitored, so that the traffic extraction and filtering are kept as simple as possible. More recently, ReSurf [10] has been proposed. Although it does a great job in overcoming the limitations of [9], it still rely on the Content-Type (as well as the Content-Size), which makes it inadequate for our needs.

3 Datasets

While per-object information is sufficient to study caching (e.g., assess caching policies and gains) or filtering (e.g., blocking unwanted objects), further processing is needed to infer human-induced HTTP transactions. This is one of the challenges that we tackle in this paper. For this purpose, we rely on two types of traces. The first is collected in a controlled test-bed where the ground truth is known and used. The second corresponds to a real traffic trace on which we apply our algorithms to appreciate the type of information that is possible to extract.

For the preliminary evaluation of our methods, we collect two ground truth traces. First, similarly to [9], [10], we build a trace (*HTTP-Alexa*) by manually visiting the top 100 most popular web sites according to Alexa ranking. In each of these sites, we randomly visit up to 10 links contained in them. We manually collect all the clicked URLs as they were reported by the browser bar. In parallel, we capture and extract HTTP requests generated by the browser. This resulted in a total list of 905 user-URLs, corresponding to 39025 browser-URLs. Second, we build a similar trace (*HTTP-GNews*) by visiting around 900 news sites promoted on Google News.

For the real data, we use the open-source traffic monitoring tool Tstat [12] to rebuild HTTP conversations. Thanks to this tool, we obtained a three-day long HTTP traffic trace, *HTTP-ISP*, which we collected at a vantage point of a commercial ISP network. The dataset contains the HTTP requests generated by

about 19,000 residential customers. In total we observed more than 190 millions of HTTP requests.

4 Content Filtering Algorithms

In this section, we introduce and evaluate our web object filtering algorithms. First, we aim at distinguishing between intentionally visited URLs, i.e., user-URLs, and automatically requested ones, i.e., browser-URLs.

4.1 Identification of user-URLs

As explained in Sec. 2, we believe that new and simpler methods must be devised to identify user-URLs. As a design choice, we assume the ability to parse only HTTP requests to reduce the complexity at the probe capturing the traffic and the amount of data to process. To this end, we propose four filtering mechanisms and compare their performance as detailed below.
1) F-Referer: This method exploits the *Referer* field, not to separate requests in different streams as done in [9], [10], but to directly pinpoint user-URLs. As shown in Fig. 1, when the URL of a web page is clicked, a sequence of HTTP requests is generated by the browser to retrieve all the embedded objects, thus generating a cascade of browser-URLs. All these requests have as a referer the starting user-URL (see Fig. 1). Therefore, for each request, this filter ignores the URL and focuses on the *referer* field, so that we are sure of capturing all the original user-URLs. However, observe that this approach captures all web objects with a hierarchical structure (e.g., JavaScript and CSS files that embed themselves other files).
2) F-Children: The nowadays trend of web page design goes towards including lots of embedded objects, which we call *children*. By counting the number of URLs seen with a given referer, it is possible to know the number of children composing the corresponding parent URL. Thus, we can filter out those URLs with a low number of children, e.g., simple objects that contain few other objects (e.g., URL-3c in Fig. 1).
3) F-Type: Similar in spirit to [7], this filter acts on URLs based on their type. However, instead of relying on the *Content-Type* field which is exposed in HTTP responses, we inspect the extensions of the objects queried by the HTTP requests: we discard URLs pointing to *.js*, *.css*, *.swf* objects.
4) F-Ad: A large amount of advertisement is embedded in nowadays web pages through the mean of iframe HTML nodes. By design, an advertisement iframe may embed several other objects itself that has as referer the advertisement page. Thus, ads likely to pass all above filters. To counter this phenomenon, we blacklist URLs pointing to known advertisement platforms using the filter provided by AdBlock [13].

In addition to these methods, we also test a time-based filter (F-Time) as proposed in [8], [9], [10], using a static time threshold.

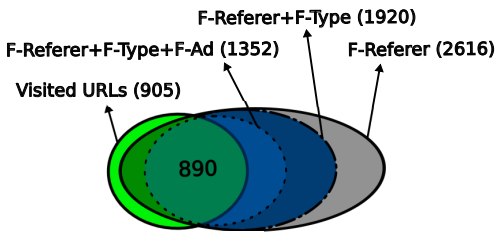

Fig. 2. Venn diagram reporting the effects of different filtering methods on our *HTTP-Alexa* dataset.

Performance on ground truth datasets We first evaluate these methods on the *HTTP-Alexa* dataset, described in Sec. 3. In Fig. 2 we graphically report the performance of filters F-Referer alone, F-Referer + F-Type and F-Referer + F-Type + F-Ad through a Venn diagram. We observe that the F-Referer method alone has a high "filtering capacity". In fact, it reduces the set of browser-URLs from the 39025 URLs in *HTTP-Alexa* to only 2616 candidate user-URLs. It has therefore a 4,4% of false positive rate (FPR)[6]. Besides, it has a high *recall*[7] (98,34%). Manually inspecting the few misses, we find that they either correspond to HTTPS visits which did not pass the referer field, or to Flash content for which no URL requests are issued for browsing. However, its precision[8] is still low (34%), which means that there is still a considerable amount of false positives. To reduce the set of false positives, we combine the F-Referer filter with the other filters with different configurations, i.e., considering different thresholds. The results are summarized in Tab. 1. Both F-Time and F-Children enhance the precision when we increase respectively the time threshold and the minimum number of children (they invalidate false positives). However, they come at the cost of decreasing the recall (invalidating valid user-URLs). On the other hand, the F-Type and the F-Ad filters let us enhance the precision, with almost no impact on the recall. In reality, the F-Ad filter removes 16 valid user-URLs (see the dark green slice that is cut by the introduction of F-Ad in Fig. 2), but these were unintentionally visited when we collected the trace[9].

We acknowledge that there is a tradeoff between recall and precision. The best combination to apply depends therefore on whether we want to capture all user-URLs or to remove all false positives. As a start, we opt for now for a most conservative approach that favors recall over precision. For the rest of the paper, we retain the F-Type coupled with F-Ad filters applied after the F-Referer method (i.e., F-Referer + F-Type + F-Ad). This guarantees 96.57% of recall, with only 1.17% of false positive rate.

[6] Number of False Positives (here 1711) over the number of negatives (here 38120)
[7] Number of URLs correctly labeled as user-URLs (true positives, here 890) over the number of user-URLs (ground truth, here 905).
[8] Number of true positives over the number of items labeled as positive by the method (here 2616).
[9] Although we acknowledge that some advertisements might interest people, we decide to skip them for now.

Method	Recall	Precision	FPR
F-Ref + F-Time (0.01s)	97.90%	37.67%	3.7%
F-Ref + F-Time (0.1s)	96.13%	41.09%	3.17%
F-Ref + F-Time (1s)	87.51%	55.15%	1.39%
F-Ref + F-Children (remove <= 1)	94.8%	43.13%	2.84%
F-Ref + F-Children (remove <= 2)	93.14%	49.76%	2.06%
F-Ref + F-Type	98.34%	46.35%	2.66%
F-Ref + F-Ad	96.57%	44.14%	2.82%
F-Ref + F-Type + F-Ad	**96.57%**	**66.41%**	**1.17%**
F-Ref + F-Type + F-Ad + F-Children (remove <= 2)	91.82%	77.08%	0.45%
F-Ref + F-Type + F-Ad + F-Children (remove <= 3)	89%	79.1%	0.29%

Table 1. Performance of different filtering combinations.

4.2 Pinpointing Interesting URLs via OSN metadata

With the goal of targeting a recommendation system, we are now interested in identifying interesting-URLs among the detected user-URLs. Indeed, manually looking we find that many of them do not correspond to URLs that might attract users' attention (e.g., the web portal of an online bank). Therefore, an extra step in the filtering is needed to go from user-URLs to *interesting-URLs*.

Finding a measure of interest is challenging since it involves human subjects and tastes. We investigate on simple heuristics that leverages online social network meta-information. We assume that interesting URLs should be rich with "social" features (e.g., share buttons). The idea is that if a web page is meant to be shared, then it might interest other people. We thus develop two methods to understand which user-URLs are "social-networks enabled". In particular we considered two approaches:
1) Active: This method uses a web scraper to actively query the URL and parse the returned HTML code looking for the presence of the OpenGraph protocol[10] [14]. If the protocol is present, the user-URL is classified as interesting.
2) Passive: This approach aims at passively detecting if a web page contains any of the well known social networks buttons. Given a user-URL X, we inspect its children, i.e., URLs having X as referer, and match them against a list of URLs necessary to load such buttons[11]. Such list was built by following the web development guidelines of several different social networks.

Performance on ground truth datasets To evaluate these methods, we test if they already work on platforms that are known to be "interesting". In particular, we test them against Google News. For this purpose, we visit 1000 URLs promoted by Google News. Applying the active and the passive methods on this trace, we find that they classify as interesting respectively 79% and 70.72% of Google News URLs. For the active method, this means that 79% of Google news URLs actually implement the OpenGraph protocol. To understand the passive method's false negatives, we manually inspect them. We find that they correspond to web sites that use custom methods to embed the social

[10] The OpenGraph protocol was developed by Facebook to help web pages getting integrated in "the social graph".
[11] Available at `http://www.retitlc.polito.it/finamore/plugins.txt`.

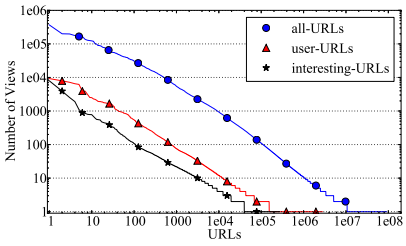

Fig. 3. Content items popularity before and after filtering (*HTTP-ISP*).

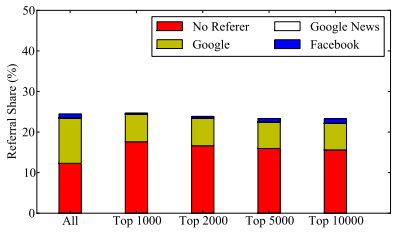

Fig. 4. Referral shares for user-URLs.

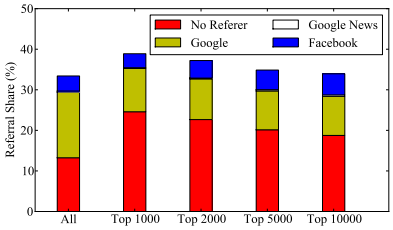

Fig. 5. Referral shares for interesting-URLs.

networks buttons and that our filter fails to identify. For instance, YouTube uses a non standard solution. However, since it is OpengGraph compliant, our active method correctly detects it. To estimate the capability of pinpointing interesting web objects, we first apply the filter chain F-Referer + F-Type + F-Ad on *HTTP-Alexa* trace to detect user-URLs; then we use our social-aware heuristics. Intuitively, the URLs Alexa ranking are less likely to be promoted than the Google News URLs, we expect them therefore to contain much less interesting URLs. This was confirmed by our both methods which gave similar results: 33% of the user-URLs were labeled as interesting-URLs using the active method, and 35% using the passive one.

These methods look promising, we use them in the rest of the paper and leave their enhancement for future work.

5 Applying the filtering on real data

In this section, we apply our filtering methods on the real data (*HTTP-ISP*) with two purposes in mind: first to rank and analyze user-URLs and interesting-URLs, and second to do some preliminary web analytics job.

5.1 Additional problems

When considering the real HTTP traffic for filtering, some additional sources of error have been detected and called for more ingenuity. In particular, we noticed

two main problems, and we propose two simple mechanisms to factor them out.

i) URLs generated by non-browser applications: Most applications use HTTP to automatically download web objects. The queried URLs are clearly not relevant, and must be ignored. We identify those URLs by inspecting the *User-Agent* field, which informs about which client application generated the HTTP request, and white-listing well-known browsers only. This discards around 15% of URL requests in our dataset.

ii) Inflated popularity induced by some users: Sometimes browsers generate multiple HTTP requests for the same content, e.g., automatically reloading a page, or downloading videos in chunks. This phenomenon inflates the popularity of some URLs. We counter this effect by counting a URL only once for each user-id.

5.2 Ranking and Classifying URLs

We first apply the F-Referer + F-Type + F-Ad filters, and study the popularity of the resulting user-URLs. Fig. 3 shows the popularity of all-URLs, user-URLs and interesting-URLs. The x-axis reports the rank of URLs, while the y-axis reports the actual number of occurrences of each URL, i.e., its popularity. Both axes are log-scale. Observe that the distribution follows a typical Zipf law, and many URLs show a low popularity: only 10,000 out of millions URLs have a popularity larger than 1000.

As expected, the user-URLs represent a tiny fraction of all the URLs. Our method detects that among the 190 million URL requests, only 1,6% correspond to actual user-URLs requests. Among these user-URLs views, only around 15% were detected as potentially interesting.

We next analyze the interesting user-URLs. We manually classify the top 1000 URLs to build a set of rules to help us having a preliminary classification of the rest of the URLs. Among these top 1000 URLs, we find that 482 are news (or blogs), 336 are services (e.g., online shops, travel engines, etc.) and 91 were videos. Extending a similar classification on the rest of the interesting user-URLs, we find that (at least) 18% correspond to videos coming from 9 different platforms (YouTube alone 15%) and 22% correspond to news from around 80 different news web sites. This observation confirms that the ISP is probably the only entity that has such a rich cross-service and wide-angle vantage point on the Internet.

Open issues We now report some of the limits observed with our filtering methods. First, for the user-URLs, although we drastically reduce the set of candidate URLs, our results still contain some false positives, which is not surprising given their relative high percentage. This is mainly due to the complex composition of web pages with a nested structure of objects. In the future, we plan to (1) test some of our other filtering methods that favor precision at the cost of recall (example the children method) as well as (2) work on enhancing our methods.

Another issue is that the returned list of the top most interesting-URLs contains popular and well-known web pages, e.g., the Google Search home page,

or the most popular news portals. This confirms on the one hand the goodness of our algorithms; on the other hand, this opens the space for smarter algorithms to pinpoint interesting-but-not-obvious-URLs.

Finally, our algorithms outperform Resurf [10] in terms of processing speed. Applying F-Ref + F-Type + F-ad on one hour of *HTTP-ISP* (19,000 users) took us around 43 minutes. This is about the same time it took Resurf to process one hour traffic of only 1,000 users. However, our main bottleneck is the F-Ad due to the large size of its catalog; F-Ref + F-Type alone take only 3 minutes to complete. We plan to optimize it as part of our future work.

5.3 Early web analytics results

Web analytics is an important tool for business and market research. It helps companies to refine the target of their advertisement campaigns, eventually giving precious hints about where to place advertisement to collect more user clicks. Most of such tools today work on a per site basis: each site has a local view on where its visits came from (referring to it). Thus, a general view is missing, and it is not clear which web platforms drive most of today's user visits. Thanks to our filtering methods, ISPs overcome this limit, reaching a cross-service view, by simply inspecting the referers associated to the detected URLs.

As a preliminary step, we start by quantifying the referral share of Google, Facebook and Google News, i.e. how many visits are driven thanks to these promotion platforms. We report our results in Fig. 4 and Fig. 5, for user-URLs and interesting-URLs, respectively. The y-axis reports the referral share in percentage over all the views in the 3 days. In the x-axis, we apply the results on different populations of URLs depending on their popularity. The "No Referer" shares correspond to URLs which had no referer (e.g., due to direct browsing, bookmarks, email, etc.).

Interestingly, a considerable amount of visits came with no referer (around 13%). This goes up to 25% for the top 1000 interesting-URLs, meaning that direct browsing is common when browsing the web. As expected, the shares of visits driven by well-known platforms are larger for interesting-URLs compared to user-URLs. However, the share of requests coming from Google is more important for unpopular objects if compared to popular ones. This is again expected since users use search engines to look for unknown content rather than popular links. However, it is surprising that despite their notoriety, the referral shares of these platforms are relatively low: Only 11% of user URLs visits came from Google (and only 16% for interesting-URLs). Observe that the rest of the shares correspond to other platforms, such as news portals and blogs. We intend to rank them as part of our future work.

Open issues One of the main issues that we figured out in this study is due to usage of HTTPS. In fact, when going from a page served over HTTPS to another served over HTTP, the standard imposes to not include the referer field [15], so that users' privacy is preserved. However, because of its importance

to marketers, many websites implement workarounds, either by using the meta referer or HTTP redirections, to still inform the landing page where the visit came from. This biases our results inflating the "No Referer" share with visits that came from HTTPS pages, and for which the referer was not passed. To get a hint on the consistency of our results concerning Google and Facebook (that both use HTTPS for browsing), we made experiments employing several popular web browsers and different operating systems. We visited URLs starting from Google Search and Facebook and check whether the referer was correctly passed. For Google, we found that the referer is always passed, but this does not always hold for Facebook, indeed we observed different behaviors depending on the browser and operating system. As a consequence, the Facebook referral shares are underestimated, but upper bounded by the "No Referer" share. We plan to investigate more this issue as part of our future work.

6 Conclusions, Open Problems and Future Work

Driven by the intuition that ISPs have a great deal of information at their disposal that can be extracted from passive observation of the traffic flowing in their networks, we propose in this paper a set of algorithms to extract popular URLs from real Internet traffic. In this study we analyzed the feasibility of the automatic extraction of useful/interesting URLs from the humongous amount of HTTP data that flows through an ISP-wide network. Other challenges must be clearly faced, and ingenuity must be used to find appropriate solutions:
i) Users' privacy: tracking users' activity is (commonly) performed by OTT providers and it has been strongly criticized recently [16]. ISPs face similar issues. They are allowed to monitor traffic for troubleshooting or even marketing purposes. Data could even be stored if properly anonymized and/or aggregated, i.e., if user's identity is irreversibly hidden. Our algorithms monitor aggregated data by design, without the need to offend users' privacy.
ii) Traffic encryption: Apart the issue discussed in Sec. 5.3, traffic encryption deeply hampers whatever traffic mining operation. This is not a negligible aspect since today more than 20% of the HTTP traffic is encrypted [17]. Mining interesting URLs in presence of encrypted traffic is another interesting research direction. However, end-customers could be involved directly, e.g., offering them personalized services (such as media curation, or parental control filters) in exchange for installing some plugins that collect clicks.
Our preliminary results suggest that the extraction of user-URLs from network traffic is a challenging task given the structure of nowadays web pages. However, "network gold mining" may represent a promising opportunity for ISPs to compete with OTTs in front of marketers and advertisers. Moreover, we believe information provided by our methodologies can be employed for a broad gamma of applications, ranging from new passive recommendation services to caching systems and parental control tools to name a few. In our ongoing efforts, we are developing each presented algorithm to run in a complete online extraction system, capable of working in a real ISP network scenario.

References

1. D. York, "What is an over-the-top (ott) application or service?." http://goo.gl/vmxVT. [Online; July 2012].
2. "Telecom italia vod." http://www.cubovision.it/.
3. "At&t cloud service." https://www.synaptic.att.com/clouduser/.
4. "Telefonica cloud service." http://www.telefonica.com/en/digital/html/digital_services/cloud.shtml.
5. "At&t joins verizon, facebook in selling customer data." http://goo.gl/FGDEp6.
6. A. Kleinman, "Verizon selling customers' cell phone data: Report." http://goo.gl/RVAEAV. [Online; Nov 2013].
7. H.-K. Choi and J. O. Limb, "A behavioral model of web traffic," in *IEEE ICNP*, (Toronto, CA), 1999.
8. P. Barford and M. Crovella., "Generating representative web workloads for network and server performance evaluation," in *ACM SIGMETRICS*, (Madison, US-WI), 1998.
9. S. Ihm and V. S. Pai, "Towards understanding modern web traffic," ACM IMC, (Berlin, DE), 2011.
10. G. Xie, M. Iliofotou, T. Karagiannis, M. Faloutsos, and Y. Jin, "Resurf: Reconstructing web-surfing activity from network traffic," in *IFIP Networking Conference*, 2013.
11. F. Schneider, B. Ager, G. Maier, A. Feldmann, and S. Uhlig, "Pitfalls in http traffic measurements and analysis," in *PAM* (N. Taft and F. Ricciato, eds.), vol. 7192 of *Lecture Notes in Computer Science*, pp. 242–251, Springer, 2012.
12. A. Finamore, M. Mellia, M. Meo, M. M. Munafò, and D. Rossi, "Experiences of Internet traffic monitoring with Tstat," *IEEE Network*, 2011.
13. "Adblock Plus." http://easylist.adblockplus.org/. [Online; July 2013].
14. "Facebook OpenGraph." http://goo.gl/2y2VN.
15. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol–http/1.1, 1999," *RFC2616*, 2006.
16. I. E. Akkus, R. Chen, M. Hardt, P. Francis, and J. Gehrke, "Non-tracking web analytics," ACM CCS, (Raleigh, US-NC), 2012.
17. A. Finamore, V. Gehlen, M. Mellia, M. Munafo, and S. Nicolini, "The need for an intelligent measurement plane: The example of time-variant cdn policies," in *IEEE NETWORKS*, (Rome, IT), 2012.