Detecting Degradation of Web Browsing Quality of Experience

Alexis Huet
Huawei Technologies, France
alexis.huet@huawei.com

Zied Ben Houidi
Huawei Technologies, France
zied.ben.houidi@huawei.com

Bertrand Mathieu
Orange Labs, France
bertrand2.mathieu@orange.com

Dario Rossi
Huawei Technologies, France
dario.rossi@huawei.com

***Abstract*—Quality of Experience (QoE) inference, and particularly the detection of its degradation is an important management tool for ISPs. Yet, this task is made difficult due to widespread use of encryption on the data-plane on the one hand so that measuring QoE is hard, and to the ephemeral properties of the web content on the other hand so that changes in QoE indicators may be rooted in changes in properties of the content itself, more than being caused by network-related events. In this paper, we phrase the QoE degradation detection issue as a change point detection problem, that we tackle by leveraging a unique dataset consisting on several hundreds thousands browsing sessions spanning multiple months. Our results, beyond showing feasibility, warn about the exclusive use of QoE indicators that are very close to content, as changes in the content space can lead to false alarms that are not tied to network-related problems.**

***Index Terms*—Changepoint detection; Web; Quality of Experience**

I. Introduction

The ultimate goal of any provider is to serve its users an enjoyable experience: this has given rise to studies defining application-specific metrics apt at capturing user quality of experience (QoE). For instance, in the context of web applications, recent work [1], [2] has shown the feasibility of accurately inferring QoE metrics, online and on a per-page visit basis, even in the presence of encrypted traffic. The inference is not limited to simple Page Load Time (PLT), but includes as well more advanced indicators of page rendering, such as SpeedIndex and variants [3], that are better correlated with user experience [4]. This is good news for network operators since they can finally closely monitor, despite encryption, the evolution in time of their users' web QoE. However, whereas most work focus on *accurately inferring such QoE metrics at a session level*, to the best of our knowledge, none of the prior work has tackled the problem of *automatically detecting relative changes* in these QoE metrics over time, which is the object of this work.

This problem, simple at first sight, hides several challenges. First and foremost, QoE degradation and network related degradations relate to each other in subtle ways. Indeed, not every network degradation induces a QoE degradation and QoE degradations are not necessarily due to network issues: e.g., when it comes to the web, other factors such as the page content, or its sources of distribution may impact user QoE even if the network is not experiencing any issue. Additionally, the proper evaluation of any degradation detection mechanism is complex in the absence of ground truth and appropriate evaluation metrics, that we also tackle in what follows.

In this paper, we consider the QoE degradation as a change-point detection problem. In particular, we instrument a Web View [5] measurement campaign to automatically collect active browsing towards a set of target webpages during several months: given the long duration of the collection, the target pages contents change during the measurement period. This allows us to collect several months worth of L7 web QoE related time series (e.g., PLT, SpeedIndex, etc.) together with other indicators of how page characteristics evolve over time. To mimic the real-world heterogeneity, we record a large set of browsing sessions from a multitude of devices using different browsers (Firefox, Chrome) and viewport configurations (SD, HD, FHD), exploiting different protocols (HTTP2, QUIC, etc.) and connected via different access technologies (ADSL, WiFi, fiber). Finally, we carefully build a synthetic ground truth by controlling network related changes while keeping all the other properties constant. We use this ground truth to evaluate our change point detection methods and oppose changes in network related features with those related to content related features. Summarizing our main contributions:

- *Dataset construction*: We propose a novel way of handcrafting network ground truth, and make our dataset available to the community [6], to provide researchers with a playground for investigating alternative methods to ours.
- *QoE degradation*: Although appealing at first for an ISP, we find that L7 web QoE indicators yield poorer detection performance of network-induced changes with respect to classic QoS indicators such as L4 transfer rate – which is due to the fact that L7 QoE indicators often correlate with changes in the content space, that are generally neglected in the literature.

The rest of the paper is organized as follows. Sec. II reviews existing change point detection methods and contextualizes them in the case of web user QoE degradation. Sec. III describes the collected datasets and the ground truth construction methodology. Next, Sec. IV presents our change point detection approach and Sec. V presents our experimental results. Finally, Sec. VI concludes the paper.

978-3-903176-31-7

II. Problem description

Network data is intrinsically a multivariate time series: ISP operations and management requires to process this continuous stream of data to detect and pinpoint few specific events that correspond to operational disruption, which possibly negatively affect the quality of experience (QoE) of their customers. A plethora of techniques exist for data processing, that can mainly be ascribed into two main areas of: *Anomaly detection*, that focuses on detection of outliers, defined as "an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data" [7]; and *Change point detection* (CPD), whose focus is on "detecting various changes in the statistical properties of time series (such as mean, variance, or spectral density)" [8]. Algorithms in the above areas are suited to capture specific types of discrepancies and are hardly interchangeable. For instance, consider two example time series from our dataset, depicted in Fig. 1. The top signal represents the Page Load Time (PLT) while the bottom signal reports the number of objects in the webpage. For the PLT series, readers can intuitively notice two change points (variance shift at t_1, mean shift at t_3, annotated with vertical blue lines) and several outliers (particularly visible when $t \in [t_1, t_3]$, annotated with black crosses). Webpage objects time series also exhibits mean and variance shifts at t_2, t_4: while the first PLT change point is unrelated to webpage changes, subsequent change points are likely related.

In this paper, we take the point of view of an ISP interested in tracking significant events for which the network it manages is responsible (to trigger appropriate actions in response), as opposed to changes that are outside its reach (e.g., as in case of content-related changes) – where by "changes" we mean variations of statistical properties of the distribution (i.e., including mean, variance, skew and higher moments) of any variable of interest (i.e., page load time, page size, etc.). We thus leverage changepoint techniques to detect events affecting a large population of individuals for a sustained duration, having thus major management consequences, as opposed to detecting punctual outliers, with a transient impact limited to few individuals.

A. Related work on CPD

It is useful to distinguish between *online* and *offline* CPD methods: while the former are useful for real-time detection, we are more interested in building a reference optimal solution and hence only consider offline methods. At high level, we can identify two families of algorithms, depending on whether (i) they exploit the *temporal dependence* among samples or (ii) merely focus on the *statistical properties of sample intervals*. Examples of the first class include Bayesian methods, with Dirichlet process Hidden Markov Multiple change point model [9] being the most popular. This model assumes a Markovian dependence from any instant t_k to the next t_{k+1}, in addition to the existence of multiple change points over time.

In our case, as samples come from many independent entities, it is reasonable to assume independence of the samples on each regime, so that algorithms in the second family are more relevant. Due to space limitation, we refer the interested reader to [10] for a structured overview of these methods. Briefly, most of this family's algorithms are based on maximizing a likelihood, with penalties to prevent the detection of too many change points. In a machine learning context, this is equivalent to *minimizing a cost function with regularization*, and many algorithms [11]–[14] exist that just differ in the selected cost function, optimization and regularization methods. We select state-of-the art method most fit to our problem in Sec. IV.

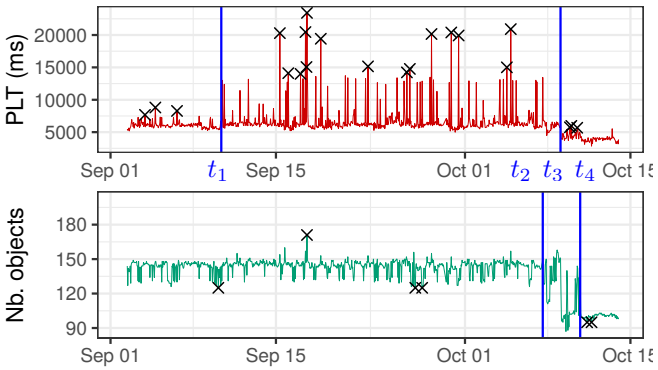


Fig. 1: Example time series from our dataset: Page Load Time (top) and number of objects in the page (bottom) in *http://lefigaro.fr* website. Vertical annotations (blue lines) are obtained with change point algorithms described later in this paper. For the sake of illustration, we mark with crosses outliers that deviate over 3σ in each interval.

B. Related work on QoE degradation

While much valuable research has targeted the issue of web quality of experience, the focus has been mainly on proposing metrics to capture the accuracy of these proposals in capturing human Mean Opinion Score (MOS) [3], [15], [16] or methods to learn these indicators from the network [2], [17], either by conducting large scale active experiments campaigns [18]–[21] or passive measurement studies involving real users [16], [17], [22]. In all the above work, the focus is on improving the inference quality *in absolute terms and for a single session.*

This work takes a complementary viewpoint and, leveraging any QoE indicator, aims at identifying points of rupture in its distribution, *in relative terms across multiple sessions*. To the best of our knowledge, despite web content evolution is regularly tracked [23] with longitudinal cartographies [24], [25] that also cover mobile web [26], and despite some of the above work [18] do measure content-related statistics, the issue that content changes can impact QoE indicators and thus be mistaken for network-related performance degradation has never been pointed out in previous literature.

III. Dataset

We use the Web View platform [27], which is described in [5], to collect web browsing session measurements from real end-user environment. We perform a large-scale measurement campaign collecting datasets (Sec.III-A) that we augment with synthetic ground truth (Sec.III-B). For repeatability purposes, we make our dataset available at [6].

A. Data collection

Overall, we collect 222k samples during 2.5 months. Each sample corresponds to a browsing event identified by (i) a *target webpage* and a *timestamp*, (ii) a set of fixed *configuration* parameters describing the browsing session, (iii) a set of *performance metrics* describing the webpage loading experiment, and (iv) a set of *page related features*.

1) Target webpage and timestamp: The dataset contains browsing sessions directed to a small but representative sample of 8 popular *target webpages* (lefigaro.fr, sina.com.cn, twitch.tv, baidu.com, yahoo.com, tumblr.com, wikipedia.org, youtube.com) pertaining to different categories (news, gaming, search, etc.). Each session is associated with a *timestamp* defined as the navigation start time recorded by the browser. Note that due to measurement budget limitations, there is a trade-off between the number of webpages to track and the diversity of configuration parameters and the measurements frequency. Since our focus is on the evolution in time, we opt for having a smaller, yet representative, number of pages and a finer-granularity evolution in time.

2) Configuration: The probe configuration is a set of fixed parameters determined before each experiment. In terms of hardware probes, we use 17 identical machines, spread in three different locations worldwide (Lannion, Paris and Mauritius Island). In terms of *network access*, the machines cover different ISPs and access technologies (ADSL, WiFi and fiber) for a total of 9 combinations. In terms of software appliances, each machine can use up to 12 browser versions, which include various versions of Chrome and Firefox. Each machine can request a different browser viewport size among the 3 most popular configurations, and can enable or disable the AdBlock plugin to emulate different user preferences. Finally, in terms of networking stack software, a total of 6 experimental combinations are obtained by configuring a specific network protocol (HTTP/1, HTTP/2 or QUIC) and caching policy (by keeping or flushing the browser cache before the experiment).

3) Performance metrics: During each experiment, browser-level and network-level metrics are collected. Browser-level performance metrics include classic metrics such as Page Load Time (PLT), Time to The First Paint (TTFP), Document Object Model time (DOM), and more advanced indicators such as Time for Full Visual Rendering (TFVR) [5]. These objective metrics are all correlated with user Quality of Experience (QoE) [3], [15] and we are interested in detecting relative changes of any of them. Without loss of generality, we consider PLT, which is the most popular metric, in what follows. The methodology can still be applied to any QoE indicator. Finally, network-level measurements include transfer rate, the share between HTTP and HTTPS protocols and the share between HTTP/1, HTTP/2 and QUIC protocols (as not all servers support HTTP/2 or QUIC yet, and can fall back to legacy protocols [28]).

4) Page features: Page-related features include a number of characteristics that only depend on the content provider, such as the page size, the number of objects of the page and the number of domains that are required to fully download the page. As early introduced, these characteristics also evolve over time and allow to build a time series whose change points can in part explain perceived performance differences tied to content-related changes.

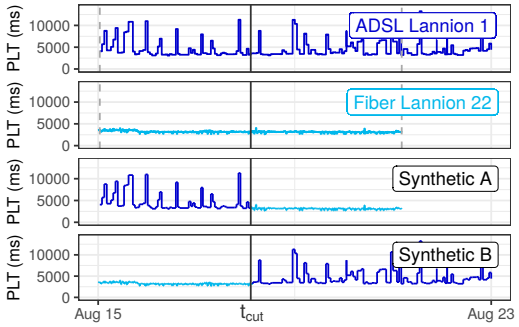


Fig. 2: Quasi-experimental methodology for building a *synthetic network ground truth*. Given a pair of time series (top), two synthetic series are computed (bottom). Solid black line indicates t_{cut}, dashed gray lines correspond to the $[a, b]$ range.

B. Ground truth

Evaluating any algorithm on data requires to compare algorithmic output with a *ground truth*, i.e., some reference solution provided by independent means. To build a reliable ground truth for network-related events, we artificially control the system conditions. We point out that the most commonly used option consists in *altering the network characteristics at experiment time*, by e.g., delaying, dropping or shaping the access bandwidth. While such a methodology is often used in networking studies, it poorly fits our use case: first, its resulting ground truth is emulated (so realistic, but not real) and only partially controlled (as those induced delays and losses are additional with respect to uncontrolled losses in the Internet); second, this would spread even further the combinatorics of the experiments (especially if combinations of the above settings are tested) and third, due to our focus on major changes, this would require to sustain the altered characteristics for a long time (trading off with the ability to test multiple combinations).

A second option, that we follow in this paper, is to *perform quasi-experiments, by pairing time series a posteriori*, in the database of globally available time series. A (semantically) similar technique was used by Akamai [29] by contrasting experiments where all parameters except one were identical, to study the relationship of video performance and user engagement. In our context, instead, we propose to build a synthetic ground truth by "cutting and sticking" pairs of time series together, where we pivot the network interface at a known timestamp as in the following steps:

- Take configurations i and j which differ only by the network interface and machine, e.g., for instance consider the case where i has slow network interface (ADSL) while j has fast network interface (fiber, WiFi),
- Multi-dimensional time series X_i spans over $[t_i, t_i']$ and X_j over $[t_j, t_j']$. Compute $[a, b] = [t_i, t_i'] \cap [t_j, t_j']$ and cut the series at $t_{\text{cut}} = \frac{a+b}{2}$,

• Extract two synthetic series with known change t_{cut} by sticking together $\{X_i\}_{t_i}^{t_{cut}}$ with $\{X_j\}_{t_{cut}}^{t'_j}$ and $\{X_j\}_{t_j}^{t_{cut}}$ with $\{X_i\}_{t_{cut}}^{t'_i}$ respectively.

This process yields two time series for which the time instant t_{cut} is objectively a change point : for the sake of clarity, this process is illustrated in Fig. 2, by depicting the PLT component of two such series, with ADSL and a fiber access technology coming from two machines both located in Lannion. Notice that change point at t_{cut} has a different semantic in series A (amelioration) vs B (degradation) and that our methodology can capture both, which is desirable to e.g., trigger alerts (B) or confirm benefits of new deployments or network optimizations (A). Overall, by pairing quasi-identical configurations, we generate a set of 2568 series[1] with synthetic network-related ground truth.

IV. Methodology

The input data to our problem is a heterogeneous collection of real-valued multivariate time series. Each point in the dataset can be indexed by a triplet (c, t, f), where c represents the current configuration, t the timestamp of the event and f the specific feature under observation. The prediction algorithm should provide an output in $\{0, 1\}$ for each triplet, where 1 indicates events of interest, a change point in our case. The ground truth introduced in the previous section can be formalized in a similar manner. From a modeling perspective, we individualize time series in our dataset by considering each of them independently. Given a configuration, the data is modelled as a piecewise independent series: each measurement can be assumed to be the result of a random variable following a certain distribution, which can only change at some unknown change points. From a networking perspective, each individualized time series comprises samples of browsing events to the same target webpage, coming from a homogeneous user population (e.g. ADSL, or WiFi, etc).

To annotate events of interest in the time series, we use as we discussed earlier offline CPD algorithms that minimize a cost function with regularization. Formally, the input of the CPD algorithm is a multivariate series $y_{1:T}$ belonging to $\mathbb{R}^{T\times F}$, with T the number of timestamps and F the number of features. The aim of the CPD algorithm is to find a number of change points K and instants of changes $0 = t_0 < t_1 < \ldots < t_{K+1} = T$, giving $K+1$ segments $(y_{t_k:t_{k+1}})_{k\in[\![0,K]\!]}$. Adopting prior characterization [30] for offline CPD algorithms, given a cost function $c(\cdot)$ which measures the cost of a segment $y_{a:b}$ (with $a \leq b \in [\![1, T]\!]$) and a penalty pen$(\cdot)$ to avoid finding too many change points, the change points are found by minimizing the following function over K and $t_1, \ldots t_K$:

$$\min_{K;t_1,\ldots t_K} \sum_{k=0}^{K} c(y_{t_k:t_{k+1}}) + \text{pen}(K, y_{1:T}). \tag{1}$$

[1]As early observed, data may not be evenly spaced in time, for which more precisely we should refer to them as *sequences* instead of *time series*. However, by abuse of language we disregard this technicality in what follows.

Overall, to define a CPD method three choices need to be made: selecting a (i) cost function, an (ii) optimization method, and (iii) a regularization method. We now specify our selection of a state-of-the art method along these three aspects.

A. Cost function

Varying the cost function allows to detect several types of changes in the underlying data distribution. The most common choice is to assume that each regime i is parametrized by a parameter θ_i, and that each sample in the regime i is distributed independently according to a density function $f(\cdot|\theta_i)$. This general $c_{i.i.d.}$ cost function is precised depending on the constraints on the parameters θ and on the choice of the density function f. For example, with $\bar{y}_{a:b}$ (respectively $\hat{\Sigma}_{a:b}$), the empirical mean vector (respectively empirical covariance matrix) on the interval $a:b$, the cost function

$$c_{L_2} = \sum_{t=a}^{b} ||y_t - \bar{y}_{a:b}||_2^2 \tag{2}$$

allows identification of changes in the mean, whereas

$$c_{\Sigma} = (b-a+1)\log\det\hat{\Sigma}_{a:b} + \sum_{t=a}^{b}(y_t - \bar{y}_{a:b})'\hat{\Sigma}_{a:b}^{-1}(y_t - \bar{y}_{a:b}) \tag{3}$$

allows identification of changes of both mean and variance.

The cost function (3) is a coherent choice for our dataset, since we can assume independence on each regime and we previously observed changes in mean and variance in Fig. 1. We point out that we do not need to assume the distribution of each regime to be Gaussian, as shown in [31], [32].

B. Optimization method

Regarding optimization method, several choices are possible, but most do not fit the settings of QoE degradation. *Exhaustive search* gives exact minimization of the parameters but it has prohibitive cost in our case as we apply the algorithm to thousands of synthetic input series. *Dynamic programming methods* give the optimal solution when the number K of change points is known, which again does not fit our settings. Whereas in our dataset the number of network change points is known (and equal to K=1), the number of content change points is not. As content changes more frequently that network outages, and as they also affect the QoE indicators, focusing only on a single change point can lead to both low recall of outages and high false alarm rate. We thus select the Pruned Exact Linear Time (PELT) [13] algorithm, which provides optimal detection of change points, with a linear computational cost under the assumption that change points do not gather around a short time interval. The latter assumption is reasonable in our settings, since major network problems affecting a large population base are expected to be infrequent (and otherwise close change points would likely be tied to the same networking problem). Thus PELT bounds the computational cost, while providing provably optimal result at the same time.

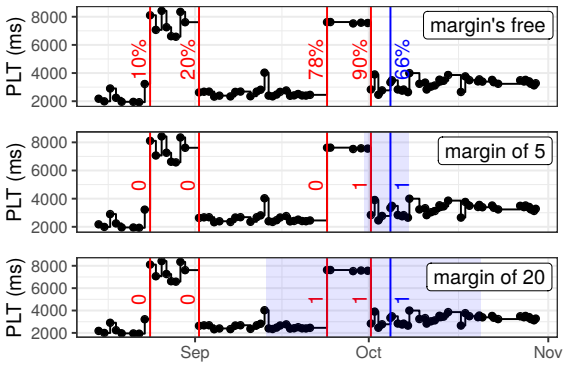


Fig. 3: Example of parameter-free vs α-margin metrics on a single series (PLT feature). For each metric, the precision (resp. recall) is defined as the average of red numbers (resp. blue number).

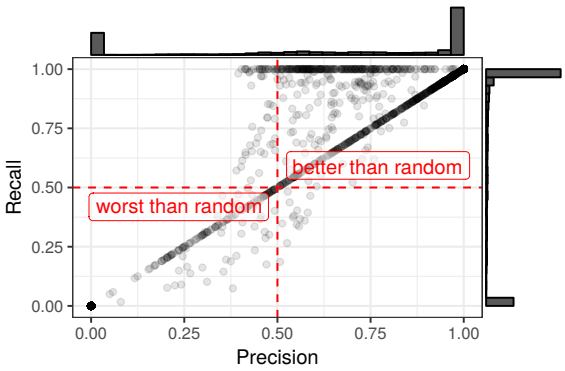


Fig. 4: Result of precision/recall for all series, for a $\beta = 10$ penalty value (PLT feature).

C. Regularization

Finally, a penalty function is necessary only when the number of change points is unknown, which is indeed our case. The most common choice is to take a linear penalty, i.e. a function proportional to the number K of detected change points [13]. However, since we want to analyze thousands of series with different lengths under the same penalty function, we account the size variability by making the penalization proportional to $\log(T)$ with T the series length, as suggested by the Bayesian Information Criterion (BIC) [33]. Thus, we define our penalty function as:

$$\text{pen}(K, y_{1:T}) = \beta K \log(T) \tag{4}$$

with $\beta > 0$ a smoothing parameter hereafter referred as *penalty value*. The selection of the penalty value is done by maximizing the agreement between the predicted and the ground truth change points according to an evaluation metric over various values of β. Such methodology has been used for example in [34] with a domain-specific evaluation metric. We perform a similar fitting of the free hyperparameter β in Sec.V, showing that β can be chosen in a large range without compromising detection accuracy.

V. Experimental results

We perform a thorough performance evaluation by exploring over 9 millions combinations (2568 series $\times$ 28 features $\times$ 128 penalty values β) by appling our CPD method (implemented in R using the `changepoint` package) to each feature independently. For each combination, we report results using different evaluation metrics: α-margin metrics (where a margin of error α is tolerated) and a parameter-free metric (a novel smoother alternative [35]). We first examplify the evaluation metrics (Sec. V-A), then illustrate the predictions obtained on the PLT series (Sec. V-B) and finally extend to a larger set of parameters, features and metrics (Sec. V-C).

A. Illustration on a single time-series

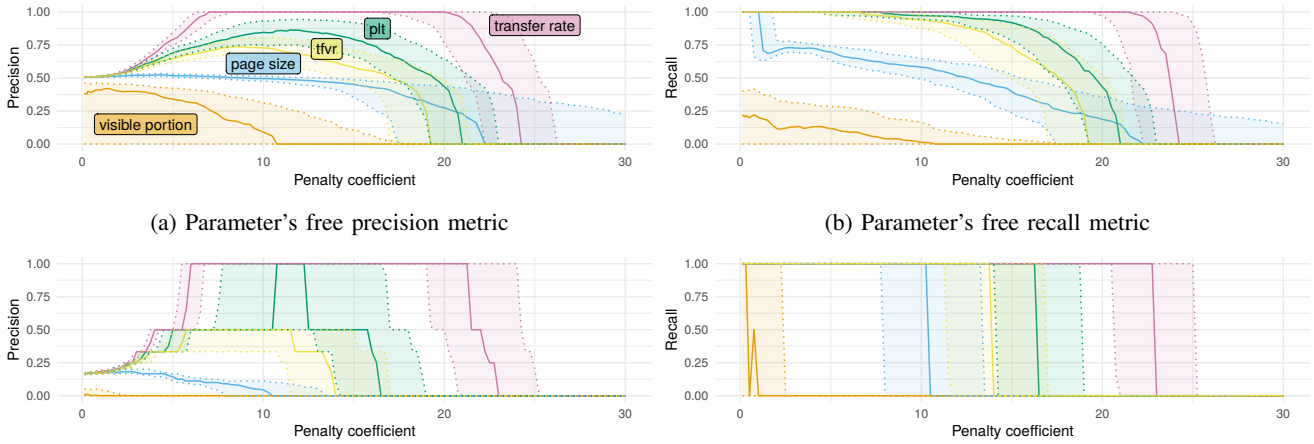
Fig. 3 presents an illustration of CPD execution (with a penalty $\beta = 10$) on the PLT time series of a popular page (YouTube), resulting in 4 predicted changes (in red) and 1 ground truth change (in blue). The performance is evaluated with three different precision/recall metrics to illustrate how each metric works. First, a commonly used metric that generalizes the precision and recall in the context of time series is represented by α-precision/recall, that allows a certain margin of error α in the time-axis. Second, a parameter free metric [35] results are presented in top plot. The latter has a simple intuitive interpretation: a *global precision around 50% indicates that predicted values are as bad as random*; in the example, global precision is 0.495 (as bad as random) and recall 0.66 (only slightly better than random). Individual precision and recall offer finer granularity: low individual precisions correspond to predicted values which are far from any true value, whereas the low individual recalls correspond to true values for which no predicted values are located nearby. Bottom plots report α-margin metrics for two values α (blue shaded region): clearly, while for $\alpha = 0$ precision and recall would be null (since the intersection between predicted and true instants is empty), for very large α both precision and recall increase to 1 by definition – so that performance evaluation is deeply affected by the value of α.

B. Result on PLT series

We aggregate all the PLT series and observe the precision vs recall scatterplot in Fig. 4, obtained fixing $\beta = 10$. For series where no change has been predicted, precision and recall are not defined but conservatively set to zero, whereas points on the diagonal correspond to cases where a single change point is predicted. Overall, we find that PLT change point prediction has a median recall and precision of 97% and 83% respectively. Moreover, as the marginals in Fig. 4 help identifying, we detect the ground truth change point in more than two thirds of the series (precision and recall of one), do not find any change for about 20% of the series (precision and recall of zero), and are worse than random for 5% of the series (precision and recall in the open interval $(0, 0.5)$). Thus, the method is able to produce a relevant output, capturing many network-related changes detected and raising few false alarms.

C. Result for all series and penalties

We now extend the analysis, contrasting in Fig. 5 detection performance for several features and penalty values. We pur-



(a) Parameter's free precision metric

(b) Parameter's free recall metric

(c) Precision metric with margin of 5

(d) Recall metric with margin of 5

Fig. 5: Precision/recall on synthetic data. Solid line for the median and external areas for 0.45 and 0.55 quantiles.

posely choose features that are either related to (i) classical network QoS performance (transfer rate), (ii) web QoE-related performance (PLT, TFVR) or to (iii) content (page size, visible portion). Clearly, since our purpose is to detect network-induced change points, we use content-related features as a "control group", which is expected to have bad detection performance. The rest of the performance metrics cover different network levels. The classic transfer rate is influenced by the ISP network but as well other networks including the content ones, whereas PLT and TFVR are application-related performance indicators that are closer to user experience. Furthermore, whereas PLT is usually considered a relatively rough L7 performance indicator, TFVR is a more sophisticated one that captures the end of the webpage rendering process and is expected to better correlate with user QoE [4], [15].

Contrasting Fig. 5-(a,b) vs (c,d), notice that parameter-free metrics [35] allow a smoother interpretation of the results: on the contrary α-margin metrics induce sharp quantized transitions as a function of the penalty, while still depending on arbitrarily set thresholds ($\alpha = 5$ in the figure). For simplicity, we focus on parameter-free metrics in what follows.

Considering control group features, we observe as expected a poor detection performance that degrades with increasing penalty. Considering performance group features (TFVR, PLT, transfer rate), we see that: (i) penalty $\beta > 0$ is beneficial in suppressing false alarms as precision initially increases; (ii) detection performances are smoothly varying on β, whose setting is thus non critical in a fairly large range; (iii) although appealing at first, tracking web QoE-related changes, compared to transfer rate, leads to more false alarms, with TFVR, albeit closer to user experience, being worse than PLT.

The last point requires a careful discussion. Our results do not imply that transfer rate is a better indicator of user QoE than PLT or TFVR on an absolute scale: rather, they state that network-induced relative changes in user QoE are better captured when change point detection methods are applied to transfer rate as opposed to PLT or TFVR. Otherwise stated, when an ISP considers a single session in isolation, application-related indicators such as PLT, and especially more sophisticated ones such as TFVR, are likely better correlated to QoE of a single user. Yet, when considering sequences of sessions, especially when contextual information on content-related changes are missing, using lower-layer network-related features seems to have more discriminative power to detect network-induced changes in the QoE of a pool of users.

Finally, note that gains are quite consistent: e.g., not only transfer rate has higher precision and recall than PLT (or TFVR), but also that its performance are stable for a wider range of penalty settings (median of precision and recall are equal 1 for $\beta \in [8, 22]$) with respect to PLT (whose precision never attains 0.9 and both precision and recall significantly degrades already for $\beta > 15$). This result has positive implication, as it lowers the barrier for ISPs to implement a simple method such the one we propose in this paper, as transfer rate measurements are already widespread.

VI. Conclusions

We present an optimal change point detection method to reveal network-induced changes of web user QoE, that we apply to a carefully collected dataset and ground truth. Albeit preliminary, our results give interesting insights for the ISP who sets out to track QoE changes using L7 metrics: While PLT and TFVR detect network changes with a high recall, they are also prone to false alarms, as pages are likely to exhibit multiple changes in the content-space, which concur in altering the user experience. We observe that classic network-QoS features like transfer rate remain better suited than more advanced QoE indicators like PLT to detect network-induced changes. As future work we plan to (i) exploit online change point detection algorithms, (ii) explicitly detect content-related changes (e.g., on page size, number of objects, etc.) to suppress false alarms and (iii) adopt multi-variate techniques (e.g., to jointly leverage PLT, TFVR and transfer rate).

REFERENCES

[1] A. Huet, Z. Ben Houidi, S. Cai, H. Shi, J. Xu, and D. Rossi, "Web quality of experience from encrypted packets," in *ACM SIGCOMM Posters and Demos*, 2019.

[2] A. Huet, A. Saverimoutou, Z. B. Houidi, H. Shi, S. Cai, J. Xu, B. Mathieu, and D. Rossi, "Revealing qoe of web users from encrypted network traffic," in *2020 IFIP Networking Conference (Networking)*. IEEE, 2020, pp. 28–36.

[3] E. Bocchi, L. De Cicco, and D. Rossi, "Measuring the quality of experience of web users," *ACM SIGCOMM Comp. Comm. Review*, vol. 46, no. 4, pp. 8–13, 2016.

[4] T. Hoßfeld, F. Metzger, and D. Rossi, "Speed Index: Relating the industrial standard for user perceived web performance to web QoE," in *IEEE QoMEX*, 2018.

[5] A. Saverimoutou, B. Mathieu, and S. Vaton, "Web View: A measurement platform for depicting web browsing performance and delivery," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 33–39, 2020.

[6] https://webqoe.telecom-paristech.fr/data/.

[7] V. Barnett, T. Lewis, and F. Abeles, "Outliers in statistical data," *Physics Today*, 1979.

[8] S. Gharghabi, Y. Ding, C.-C. M. Yeh, K. Kamgar, L. Ulanova, and E. Keogh, "Matrix profile VIII: Domain agnostic online semantic segmentation at superhuman performance levels," in *IEEE International Conference on Data Mining (ICDM)*, 2017.

[9] S. I. Ko, T. T. Chong, P. Ghosh *et al.*, "Dirichlet process hidden Markov multiple change-point model," *Bayesian Analysis*, vol. 10, no. 2, pp. 275–296, 2015.

[10] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, 2020.

[11] M. Lavielle and E. Moulines, "Least-squares estimation of an unknown number of shifts in a time series," *Journal of time series analysis*, vol. 21, no. 1, pp. 33–59, 2000.

[12] B. Jackson, J. D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, and T. T. Tsai, "An algorithm for optimal partitioning of data on an interval," *IEEE Sig. Proc. Letters*, vol. 12, no. 2, pp. 105–108, 2005.

[13] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.

[14] K. Frick, A. Munk, and H. Sieling, "Multiscale change point inference," *Journal of the Royal Statistical Society: Series B*, vol. 76, no. 3, pp. 495–580, 2014.

[15] D. N. da Hora, A. S. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the gap between QoS metrics and Web QoE using above-the-fold metrics," in *International Conference on Passive and Active Network Measurement*, 2018, pp. 31–43.

[16] F. Salutari, D. Da Hora, G. Dubuc, and D. Rossi, "Analyzing Wikipedia users' perceived quality of experience: A large-scale study," *IEEE Transactions on Network and Service Management*, 2020.

[17] M. Trevisan, I. Drago, and M. Mellia, "Pain: A passive web performance indicator for isps," *Computer Networks*, vol. 149, pp. 115–126, 2019.

[18] A. Saverimoutou, B. Mathieu, and S. Vaton, "A 6-month analysis of factors impacting web browsing quality for qoe prediction," *Computer Networks*, vol. 164, p. 106905, 2019.

[19] A. S. Asrese, S. J. Eravuchira, V. Bajpai, P. Sarolahti, and J. Ott, "Measuring web latency and rendering performance: Method, tools, and longitudinal dataset," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 535–549, 2019.

[20] M. Rajiullah, A. Lutu, A. S. Khatouni, M.-R. Fida, M. Mellia, A. Brunstrom, O. Alay, S. Alfredsson, and V. Mancuso, "Web experience in mobile networks: Lessons from two million page visits," in *The World Wide Web Conference*, 2019, pp. 1532–1543.

[21] V. Mancuso, M. P. Quirós, C. Midoglu, M. Moulay, V. Comite, A. Lutu, Ö. Alay, S. Alfredsson, M. Rajiullah, A. Brunström *et al.*, "Results from running an experiment as a service platform for mobile broadband networks in europe," *Computer Communications*, vol. 133, pp. 89–101, 2019.

[22] M. Trevisan, D. Giordano, I. Drago, M. M. Munafò, and M. Mellia, "Five years at the edge: Watching internet from the isp network," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 561–574, 2020.

[23] D. Fetterly, M. Manasse, M. Najork, and J. L. Wiener, "A large-scale study of the evolution of web pages," *Software: Practice and Experience*, vol. 34, no. 2, pp. 213–237, 2004.

[24] T. Callahan, M. Allman, and V. Paxson, "A longitudinal view of http traffic," in *International Conference on Passive and Active Network Measurement*, 2010, pp. 222–231.

[25] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig, "Web content cartography," in *ACM SIGCOMM Internet measurement conference*, 2011, pp. 585–600.

[26] T. Johnson and P. Seeling, "Desktop and mobile web page comparison: Characteristics, trends, and implications," *IEEE Communications Magazine*, vol. 52, no. 9, pp. 144–151, 2014.

[27] https://webview.orange.com.

[28] M. Varvello, K. Schomp, D. Naylor, J. Blackburn, A. Finamore, and K. Papagiannaki, "Is the web HTTP/2 yet?" in *International Conference on Passive and Active Network Measurement*, 2016, pp. 218–232.

[29] S. S. Krishnan and R. K. Sitaraman, "Understanding the effectiveness of video ads: a measurement study," in *ACM IMC*, 2013, pp. 149–162.

[30] C. Truong, L. Oudre, and N. Vayatis, "ruptures: change point detection in Python," *arXiv preprint arXiv:1801.00826*, 2018.

[31] V. Jandhyala, S. Fotopoulos, I. MacNeill, and P. Liu, "Inference for single and multiple change-points in time series," *Journal of Time Series Analysis*, vol. 34, no. 4, 2013.

[32] M. Lavielle, "Detection of multiple changes in a sequence of dependent variables," *Stochastic Processes and their applications*, vol. 83, no. 1, pp. 79–102, 1999.

[33] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[34] T. D. Hocking, G. Schleiermacher, I. Janoueix-Lerosey, V. Boeva, J. Cappo, O. Delattre, F. Bach, and J.-P. Vert, "Learning smoothing models of copy number profiles using breakpoint annotations," *BMC bioinformatics*, vol. 14, no. 1, p. 164, 2013.

[35] (2020) Detecting degradation of web browsing quality of experience (extended version). https://webqoe.telecom-paristech.fr/data/.