Narrowing the gap between QoS metrics and Web QoE using Above-the-fold metrics

Diego Neves da Hora[1], Alemnew Sheferaw Asrese[3], Vassilis Christophides[2], Renata Teixeira[2], and Dario Rossi[1]

[1] Telecom Paristech – first.lastname@telecom-paristech.fr
[2] Inria – first.lastname@inria.fr
[3] Aalto University – alemnew.asrese@aalto.fi

Abstract. Page load time (PLT) is still the most common application Quality of Service (QoS) metric to estimate the Quality of Experience (QoE) of Web users. Yet, recent literature abounds with proposals for alternative metrics (e.g., Above The Fold, SpeedIndex and their variants) that aim at better estimating user QoE. The main purpose of this work is thus to thoroughly investigate a mapping between established and recently proposed objective metrics and user QoE. We obtain ground truth QoE via user experiments where we collect and analyze 3,400 Web accesses annotated with QoS metrics and explicit user ratings in a scale of 1 to 5, which we make available to the community. In particular, we contrast domain expert models (such as ITU-T and IQX) fed with a single QoS metric, to models trained using our ground-truth dataset over multiple QoS metrics as features. Results of our experiments show that, albeit very simple, expert models have a comparable accuracy to machine learning approaches. Furthermore, the model accuracy improves considerably when building per-page QoE models, which may raise scalability concerns as we discuss.

1 Introduction

The Web remains one of the dominant applications in the Internet. Originally designed to deliver static contents such as text and images, it evolved to serve very dynamic and complex content: it is not uncommon for modern pages to include hundreds of objects and dozens of scripts, placed at different servers hosted in different domains [11]. Given this complexity, the Web architecture and protocol landscape evolved as well, aiming at more efficient operation and to enhance the end user QoE: the introduction of Content Delivery Network (CDN) and different protocols such as HTTP2 [7], SPDY [16], QUIC [19] are some of the efforts in this regard.

Measuring the impact of different network and Web browsing configurations on Web browsing performance is essential to enhance user satisfaction. The metric most commonly used to measure the performance of Web browsing has been the Page Load Time (PLT), which holds true for both research [13, 26, 21, 27, 25] and industry [1, 2, 4]. Recent studies [10, 3, 8, 18, 24, 15], however, started to

question the relevance of using PLT to measure quality of user experience. The main skepticism is that whereas PLT measures the precise time at which the page finishes loading, the experience of the user depends on the whole process up to that time and the rendering time at the browser. As such, a number of alternative metrics, which we review in Section 2.1, such as the Above-the-Fold (ATF) time [10], SpeedIndex [3], Object/ByteIndex [8] and PerceptualSpeedIndex [15] have been proposed to bridge this gap.

The approach adopted by the measurement community for computing metrics like ATF time and SpeedIndex requires taking a series of screenshots of the Webpage loading progress and post-processing the captured frames. Unfortunately, this approach is computationally intensive, which makes these metrics complex to measure [15]. Our first contribution (presented in Section 3) is to propose a **tractable method to estimate the ATF metric, and offer an open-source implementation as a Chrome extension** [5].

Still, to date the relationship between this class of objective metrics and the user subjective feedback (e.g., via explicit ratings summarized with Mean Opinion Score (MOS)) remains to be elucidated. Indeed, while models mapping PLT to an estimated MOS do exist [17, 14] (see Section 2.2), to the best of our knowledge, extensions of these models to leverage these new metrics are still lacking. Recently, Gao et al. [15] evaluated machine learning models that use these new metrics as features to forecast A/B test results, where users are asked to compare two Webpages loading side-by-side and identify which one loads faster. Although Gao et al.'s work [15] represents an important step in the right direction, A/B tests are a special case: i.e., we still miss an answer to the more general question of how to estimate QoE of a single page a given user visits.

In this paper, we thoroughly investigate a mapping $f(\cdot)$ between user QoE, expressed in terms of subjective MOS, and some QoS factor x that represents objective measured properties of the browsing activity. In particular, we are interested in cases where x can be any combination of the above objective metrics and where the mapping $f(\cdot)$ is either defined by a domain expert (e.g., according to popular models like ITU-T [17] or IQX [14]) or data-driven models learned using classic machine learning algorithms (e.g., SVR regression, CART trees).

The other main contribution of this paper (presented in Section 4) is to **perform a thorough assessment of expert models (ITU-T [17], IQX [14], etc.) and contrast them to models learned from the data using different machine learning algorithms**, which our investigation finds to have surprisingly comparable accuracy performance. Our analysis relies on a dataset with 3,400 Web browsing sessions where users explicitly rated the quality of the session. This dataset extends our previous effort [9] and we make available to the community [28]. We conclude that expert models for Web QoE can easily accommodate new time-related metrics beyond PLT, and that their accuracy is comparable to that of data-driven models. Still, we gather that there is room for improvement, as a single expert model is hardly accurate for the wide variety of Web pages. At the same time, while we find that per-page models have superior forecast performance, the approach is clearly not scalable, which opens new

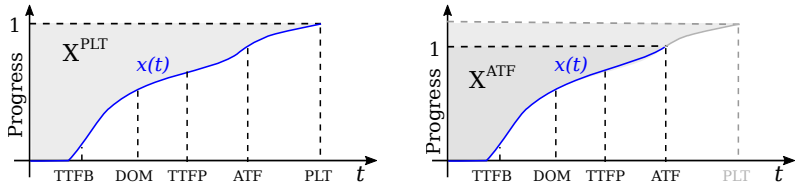


Fig. 1: Illustration of time-instant (x-axis labels) and time-integral metrics (shaded surface). The time horizon of the time-integral metrics can be limited to, e.g., (a) PLT or (b) Above-the-Fold time instants.

interesting research questions for the community to address, which we discuss in Section 4.4. We conclude in Section 5.

2 Background and related work

This section first discusses the existing metrics that aim to capture Web QoS, which we build on to define a practical method to infer the ATF time in Section 3. Then, it presents the existing models to estimate Web QoE from these QoS metrics, which we evaluate in Section 4.

2.1 Web QoS metrics

The Web browsing process is complex with the request, download, and rendering of all objects making up a Webpage. Hence, measuring when the page has finished loading from the user's perspective is challenging. The literature introduces two classes of objective QoS metrics, which we exemplify with the help of Fig. 1.

Time instants. The time to load a Web page has a number of components, such as the time at which the first byte is received (TTFB), the time at which the first object is painted (TTFP) by the browser, the parsing of the Document Object Model (DOM), to the complete download (PLT, that we measure using the onLoad browser event) or the rendering of the full page (VisualComplete). We notice that whereas network-related time-instant metrics (e.g. TTFB, DOM, PLT) are easy to measure, rendering-related metrics (e.g. TTFP, VisualComplete) are harder to define across browsers [20]. An interesting metric proposed by Google in this class is represented by the ATF time [10], defined as the time at which the content shown in the visible part of the Webpage is completely rendered. Albeit interesting, the ATF metric is neither available in Webpagetest[4], nor defined in W3C's navigation timing specifications.[5] This omission is possibly due to the fact that the ATF time is significantly more complex to measure, as it requires taking screenshots during the rendering process and a post-processing stage of the captured frames. One of our contributions is to propose a practical way to approximate the ATF time, as well as provide an open source implementation.

[4] `https://www.webpagetest.org/`
[5] `https://www.w3.org/TR/navigation-timing/`

Time integrals. Another class of metrics recognizes that a single time instant hardly captures all the complexity of interactions between the user and the rendering process of the page. Instead, this class integrates the loading time over all events of a given type throughout the evolution of a page progress. Following Google's original SpeedIndex (SI) [3] definition, a number of generalizations have been proposed in the literature [8, 15]. Metrics in this class fit the general form:

$$X^{end} = \int_0^{t_{\text{end}}} (1 - x(t))dt \tag{1}$$

where X^{end} is the value of the metric, t_{end} indicates an event considered as time horizon and $x(t) \in [0, 1]$ is the completion rate at time t. In particular, SpeedIndex (SI) [3] measures $x(t)$ as the visual progress using mean pixel histogram difference computed until the VisualComplete time. ObjectIndex (OI) and ByteIndex (BI) [8] use the percentage of objects (and bytes) downloaded until the PLT. Finally, PerceptualSpeedIndex (PSI) [15] uses Structural Similarity to measure the visual progress $x(t)$ and cut the time horizon at either the PLT, or at an arbitrary time earlier than PLT.

One interesting question is how to select t_{end}. A previous A/B study [15] showed two pages rendering processes side by side, and asked users to click on the page that completed faster: the best predictor uses the Time to Click as t_{end}, which considerably improves PSI accuracy in estimating user QoE [15]. Our experiments show that setting t_{end} with the ATF time is a good option, and our method to compute the ATF time enables measuring it during normal user browsing (i.e., without requiring user intervention).

2.2 Web QoE models

The metrics introduced in the previous section are measurable automatically from the browser (even though those involving rendering are fairly complex to compute). These metrics, however, may not directly capture the user experience (or QoE), which is often measured explicitly by an opinion score and summarized with the MOS. There are two main approaches for mapping of QoS metrics into MOS: *expert models*, where domain experts specify a closed form function and use MOS data to fit model parameters, or *machine learning models*, where MOS data is used to train the model.

Expert models. Two well established [22], models of Web QoE are the ITU-T recommendation model [17] and the IQX [14] hypothesis. The ITU-T model follows the Weber-Fechner Law and assumes that the user QoE has a logarithmic relationship with the underlying QoS metric. The model is in the form:

$$QoE(x) = \alpha \log(x) + \gamma, \tag{2}$$

where x is the QoS metric (typically, PLT) and with α, γ parameters. The ITU-T models are derived for three different contexts (fast, medium, and slow networks)

with a different minimum and maximum session time for the different contexts so that QoE $\in [1, 5]$.

Alternatively, the model based on the IQX hypothesis [14] postulates an exponential interdependency between QoE and QoS metrics. The idea of the model is that if the QoE is high, a small variation in the underlying QoS metric will strongly affect the QoE. Instead, a degradation in QoS metric will not lower QoE as much if the overall QoE is already bad. Under IQX, for a given change in QoS metric the change of QoE depends on the current level of QoE as:

$$QoE(x) = \alpha e^{-\beta x} + \gamma \tag{3}$$

where x is a QoS metric and with α, β, γ parameters. We evaluate both logarithmic and exponential models in Section 4.

Machine learning. While machine learning algorithms have been used to model QoE for VoIP [12], video streaming [6] or Skype [23], its application to Web browsing is still lacking. One marked exception is the work by Gao et al. [15], where authors formulate a ternary classification task (i.e., A is faster, B is faster, none is faster) and employ Random Forest and Gradient Boosting ML techniques with QoS metrics such as those described in Section 2.1 as input features. In this paper, we focus on a more difficult task, formulated as a regression problem in the support $MOS \in [1, 5] \subset \mathbb{R}$, and additionally contrast ML results to those achievable by state of the art expert models.

3 Approximating the ATF time

One way to calculate the ATF time is to monitor the page rendering process and identify when the pixels on the visible part of the page, also known as the *above-the-fold* part, stop changing. This can be done by monitoring the individually rendered pixels (or histograms of the rendering) and detecting when they stabilize. This approach, however, is processing intensive and difficult to apply in the wild, as the overhead may impair user experience. Webpages also contain visual jitter due to, for example, layout instabilities or carousel elements [15], making it harder to detect the ATF time using pixel comparison methods.

Methodology. We propose a method to approximate the ATF time from the browser itself without requiring image processing. We leverage the browser's ability to determine the position of objects inside a fully rendered page and the recorded loading times of HTTP requests. Our method works as follows. First, we detect all the elements of the Webpage and the browser window size. Then, we trace loading time and resource type for all HTTP requests, and determine which objects are rendered above-the-fold. To do so, we use simple heuristics to classify resource types between images, JavaScripts (JS), CSS, HTML, etc. For objects that are directly rendered (e.g., of the image class), the coordinates make it obvious whether they are, at least partly, above-the-fold. For objects for which we have no direct indication whether they are used for rendering

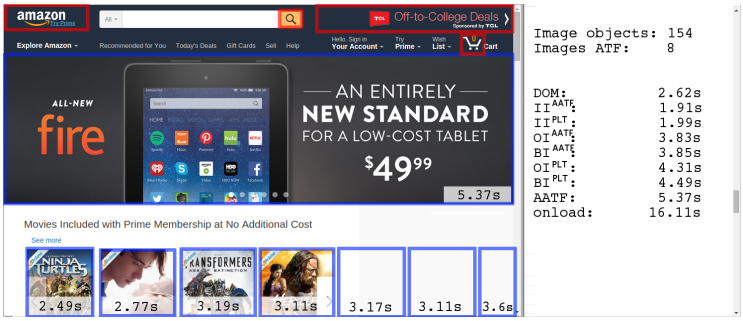


Fig. 2: Extension example: *Time-instant* metrics show that whereas DOM loads at 2.62s, all objects above the fold are rendered on or before AATF=5.37s and then the page finishes loading at PLT=16.11sec. By definition, *Time-integral* metrics are even shorter BI^{AATF}<BI^{PLT}<AATF, hinting that PLT may be significantly off with respect to timescales relevant to the user perception.

(e.g., styles that are defined through CSS; visual changes generated by JS), we conservatively assume they are required for rendering above-the-fold content. More formally, denoting with T_o the loading time of object o, and letting $\mathcal{I}$ be the set of all images, $\mathcal{I}_{ATF}$ the subset of images whose coordinates are at least partially above-the-fold, $\mathcal{J}$ the set of all JavaScript HTTP requests and $\mathcal{C}$ the set of all CSS requests, we calculate the Approximate ATF (AATF) time as:

$$AATF = \max_{o}\{T_o | o \in \mathcal{J} \cup \mathcal{C} \cup \mathcal{I}_{ATF}\} \tag{4}$$

We stress that AATF should not be considered as a replacement metric for ATF: to that extent, it would be necessary to comprehensively validate AATF against pixel-based measurements of ATF, which we leave for future work. At the same time, our experiments indicate that AATF has a good discriminative power as it helps ameliorate forecasts of user MOS, and as such has value on its own.

Implementation. We implemented the method to approximate the ATF time as an open-source Chrome extension [5]. The script executes after the `onLoad` event triggers. We use jQuery to detect visible DOM objects. For each object, we detect its position and dimensions on the page. We use this information alongside the dimension of the browser window, which we obtain using JavaScript, to determine which DOM objects are visible and above-the-fold. We use the `Window.performance` API to obtain the name, type, and timing information about the resources loaded in the page. We compare the `src` field of DOM object to the url of HTTP request to match HTML objects to its corresponding timing information. Finally, we calculate the AATF time using (4). Fig 2 shows and comments an example of the results from the extension applied when browsing the Amazon Webpage. It can be seen that only 8 of the 154 images are located above-the-fold (circled in blue in Fig 2), with a significant difference between PLT, ATF and derived metrics.

Approximations and limitations. As in any real-world deployment, we find a number of technicalities which complicates the process of detecting the resources located above the fold. For instance, some Webpages contain sliding images which keep rotating in the above-the-fold area. Additionally, there are cases where images happen to be above-the-fold but also overlap, so that some of them are not actually visible. For the sake of simplicity, we assume that all the images are visible for the AATF time calculation, which makes a conservative approximation. Also, in our current implementation, we consider images but do not take into account other multimedia object types (e.g., Flash) that may be relevant and that we leave for future work.

In some cases, we find image HTTP requests that do not match to any known HTML object. This issue happens, for example, when the background image of buttons is put into place using CSS (circled in red in Fig 2). Although we cannot reliably detect if these "unmatched" images are above or below the fold, we can still calculate the AATF time either considering that those images are always "above" (i.e., which upper bounds the AATF time) or "below" the fold (i.e., a lower bound). Our investigation reveals that whereas the PLT vs AATF difference is significant, these low-level details have no noticeable impact on the AATF time computation (not reported here for lack of space).

4 Modeling WebQoE

In this section, we thoroughly explore how Web QoS metrics relate to user QoE. We detail the dataset used in this analysis, explore how well expert models can predict QoE, and to what extent machine learning approaches present an advantage in comparison to expert models.

4.1 Dataset

To assess the impact of application QoS on QoE, we extend our previous experiment on measuring Web user experience [9]. We gather 8,689 Web browsing sessions from 241 volunteers, that we make available at [28]. During each Web browsing session, a script guides the user to select one Webpage from a list, open the page on the browser, and provide QoE feedback using the Absolute Category Rating (ACR) (from 1-Bad to 5-Excellent). For lack of space, we refer readers to [9] for a detailed presentation of our experimental setup.

In this work, we focus on 12 non-landing Webpages from the Alexa top 100 popular pages in France, with diverse page size (0.43–2.88 MB), number of objects (24–212), and loading times varying by over one order of magnitude. Since we rely on volunteers to obtain user opinion scores, we employ basic dataset sanitization techniques. First, we remove from the dataset all samples with no user rating or where the page failed to load completely. Then, we remove users who failed to complete at least 10 reviews. We keep 224 out of the original 241 users, and 8,568 out of 8,689 reviews. Finally, we restrict our analysis to only 12

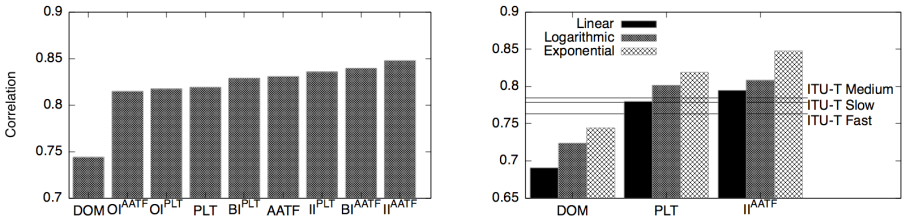


Fig. 3: Expert models: Impact of (a) explanatory QoS metric x for the $f(\cdot)$ =IQX hypothesis and (b) combined impact of metric x and mapping function $f(\cdot)$

out of the 25 original Webpages comprising a significant number of reviews and experimental conditions, which leaves us with 3,400 user ratings.

We obtain MOS values by averaging the opinion score of a Webpage for specific user groups. These groups are defined based on the distributional characteristics of the input QoS metric x, whose impact on MOS we are interested to assess. We grouped the user ratings of each page in 6 bins, specifically at every 20^{th} percentile of metric x until the 80^{th} percentile, and further break the tail in two bins each representing 10% of the population. All volunteers used identical devices during the experiments. The models we obtain implicitly assume the experimental conditions observed in this dataset such as the screen size, performance expectation, and device capabilities.

4.2 Expert models

Application metrics. To assess how well a function $f(\cdot)$ applied to a QoS metric x correlates with MOS, we consider the following time-instants: (i) the time to load the DOM, (ii) the time to load the last visible image or other multimedia object AATF and (iii) the time to trigger the onLoad event PLT. We additionally include time-integral metrics with either an AATF time or PLT time-horizon: specifically, we consider (iv) two ByteIndex BI^{AATF}<BI^{PLT} metrics, where $x(t)$ express the percentage of bytes downloaded at time t, and (v) two ObjectIndex OI^{AATF}<BI^{PLT} metrics, where $x(t)$ counts the percentage of objects downloaded at time t. Finally, we define (vi) two ImageIndex II^{AATF}<II^{PLT} metrics, where $x(t)$ only considers the size of objects of the image class, to purposely exacerbate the prominent role of images in the visual rendering of a page.

Figure 3-(a) assesses the impact of the nine selected QoS metrics on QoE, using the IQX model. We observe that, apart from DOM, all metrics show a strong (> 0.8) Pearson correlation with MOS. Specifically, we see that counting bytes (BI) and especially image bytes (II) is more valuable than counting objects (OI). Additionally, results confirm the importance of evaluating time-integrals by narrowing their time-horizon before the PLT (as suggested by Gao et al. [15]), confirming the importance of estimating the ATF time (as proposed in this paper). Overall, the metric with best correlation to MOS is II^{AATF} (0.85), with PLT ranking seventh (0.81). These results confirm the soundness of using the AATF time as proxy of user-perceived page loading time [24].

Mapping functions. We use three functions to map QoS metrics to user QoE: specifically, a linear $\mathbf{1}(\cdot)$ function, a logarithm function of the form of (2), and an exponential function of the form of (3). While the rationale behind (2) and (3) come from the Weber-Fetchner law and the IQX hypothesis, we stress that many works still *directly* compare PLT statistics, which is analogous to a simplistic linear mapping. We carefully calibrate the model parameters using the non-linear least squares Marquardt-Levenberg algorithm. In Figure 3-(b) we contrast how these different mappings correlate to QoE for a relevant subset of the QoS metrics: specifically, we select the most widely used metric (PLT) as well as those metrics exhibiting the worst (DOM) and the best (II^{AATF}) correlation with user QoE. We also compare results with the reference obtained by default ITU-T models for slow/medium/fast network conditions using the PLT metric.

Among the default ITU-T models, the model for medium networking conditions shows the stronger correlation to QoE in our dataset. This can be explained by users' expectation of network performance, since the experimental network conditions mirror that of Internet Web access. It is worth noting that the uncalibrated ITU-T medium model is still better than a linear mapping of PLT to QoE. We observe across all metrics in our dataset that the exponential mapping is superior to logarithmic, which is in turn superior to simply using a linear mapping to estimate QoE. It is easy to observe that our proposed metrics based on the AATF time (particularly, II^{AATF}) consistently yields the strongest correlation with MOS, across all functions.

4.3 Machine Learning

We evaluate different machine learning techniques to learn regression models that predict user QoE. Note that the learned function $f(\cdot)$ maps a vector $\underline{x}$ to MOS, compared to the expert models where x is a scalar metric. We evaluate the performance of three state-of-the-art machine learning algorithms: Support Vector Regression (SVR), Classification And Regression Tree (CART), and AdaBoost with CART (BOOST) implemented using the sci-kit learn Python module.

Parameter tuning. We tune the hyper-parameters of the ML algorithms using grid optimization. Namely, we select the best combination of parameters $\epsilon \in [10^{-2}, 1]$, $\gamma \in [10^{-3}, 10]$ and $C \in [1, 10^4]$ for SVR, minimum number of samples per leaf $\in [1, 10]$ and tree depth $\in [1, 10]$ for CART and BOOST, and number of boosted trees $\in [10, 10^3]$ for BOOST. Grid optimization outputs $\epsilon = 0.3$, $\gamma = 10^{-3}$, and $C = 10^4$ for SVR, and suggests 4 samples per leaf and tree depth of 2 for both CART and BOOST, and 10^2 trees for BOOST.

Feature selection. We employ three strategies for building predictors using different sets of features from our dataset. The first baseline strategy considers as features the 9 *raw* metrics defined in Section 4.2. The second strategy feeds the ML model with the output of the 3 expert models computed on the 9 raw metrics, for an *extended set* of 27 features (notice that since one mapping func-

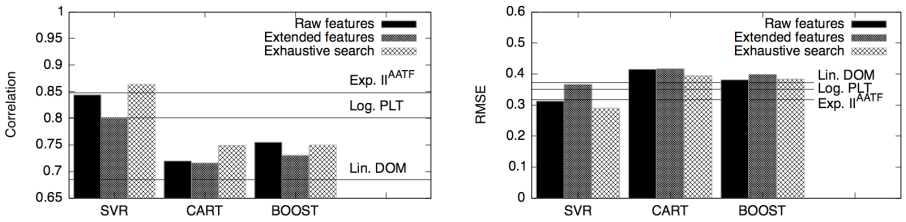


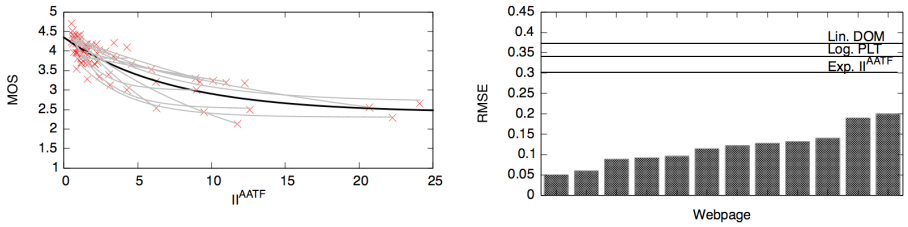
Fig. 4: Comparison of ML algorithm using different feature sets against reference expert models, for correlation and RMSE metrics

tion is linear, there are 18 additional features beyond the raw ones). Finally, as performance upper bound, we perform an *exhaustive search* of feature subsets from the extended set, to select the combination that minimizes the Root Mean Squared Error (RMSE) of the predictor. The selected combinations include few features (3–5 out of 9) that vary across ML algorithms, although the sets consistently include II^{PLT} (all algorithms) AATF and II^{AATF} (all but one).

Results. We evaluate ML predictors using leave-one-out cross-validation. Figure 4 shows the (a) correlation and (b) RMSE between MOS and the ML model, for the full set of algorithms and feature selection strategies. We also report, as a reference, the performance of the best expert model (exponential, II^{AATF}), a traditional model (logarithmic, PLT), and the worst expert model (linear, DOM). Similar considerations hold for both correlation (the higher the better) or RMSE (the lower the better): BOOST presents a small advantage over CART trees, although SVR outperforms them both. Yet, the picture clearly shows that SVR results are on par with the best expert model, with a small advantage arising in the optimistic case of an exhaustive search for feature selection.

4.4 Discussion

We believe that there is further room for improvement. Notably, we argue that, due to the variety of Webpages, the attempt to build a one-size-fit-all model is doomed to fail. To show this, we report in Figure 5 an extreme example, where (a) we build a model per Webpage and (b) contrast the RMSE results in the per-page vs. all-pages model cases: it is immediate to see that RMSE drastically decreases under fine grained models – the gap is comparably larger than what could be reasonably achieved by further refining the metrics definition, or by the use of more complex expert (or learned) models. Clearly, given the sheer number of Webpages, it would be highly unrealistic to attempt to systematically build such fine-grained models. At the same time, we believe that due to the high skew of Web content, it would be scalable to (i) build per-page models for only very popular pages (e.g. the top-1000 Alexa) and (ii) build per-class models for the rest of pages, by clustering together pages with similar characteristics. Whereas our dataset currently includes few pages to perform a full-blown study, we believe



(a) Black: one model for all pages, Gray: one model per page

(b) Lines: one model for all pages, Bars: one model per page

Fig. 5: Discussion: one model for all pages vs. one model per page

that crowdsourcing efforts such as Gao et al. [15] and systematic share of dataset can collectively assist the community to achieve this goal.

5 Conclusions

This paper narrows the gap between QoS and QoE for Web applications. Our contributions are, first, to motivate, define and implement a simple yet effective method to compute an Approximated ATF time (AATF) [5], which is also useful to narrow the time-horizon of time-integral metrics [15]. Second, we carry on a large campaign to collect a dataset of nearly 9,000 user subjective feedback, which we use for our analysis and make available to the community [28]. Finally, we systematically compare expert vs. data-driven models based on a set of QoS metrics, which include the ATF time approximation and variants. In a nutshell, our results suggest that whereas considering PLT metric with linear mapping should be considered a discouraged practice. Using (i) an exponential IQX mapping, (ii) over time-integral metrics considering ByteIndex progress of image-content only, and (iii) narrowing the time-horizon to the AATF time, provides a sizeable improvement of Web QoE estimation. Finally, we found that (iv) calibrated expert models can provide estimations on par with state-of-the-art ML algorithms.

Acknowledgments

We are grateful to our shepherd Mike Wittie and to the anonymous reviewers, whose useful comments helped us improving our work. This work has been carried out at LINCS (`http://www.lincs.fr`) and benefited from support of NewNet@Paris, Ciscos Chair "Networks for the Future" at Telecom ParisTech and the EU Marie curie ITN program METRICS (grant no. 607728).

References

1. `https://googlewebmastercentral.blogspot.fr/2010/04/using-site-speed-in-web-search-ranking.html`.

2. http://googleresearch.blogspot.fr/2009/06/speed-matters.html.
3. https://sites.google.com/a/webpagetest.org/docs/using-webpagetest/metrics/speed-index.
4. Alexa Internet Inc. http://www.alexa.com.
5. Approximate ATF chrome extension. https://github.com/TeamRossi/ATF.
6. C. G. Bampis and A. C. Bovik. Learning to predict streaming video qoe: Distortions, rebuffering and memory. *CoRR*, abs/1703.00633, 2017.
7. M. Belshe, R. Peon, et al. Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540, 2015.
8. E. Bocchi, L. De Cicco, et al. Measuring the quality of experience of web users. *ACM SIGCOMM CCR*, 2016.
9. E. Bocchi, L. De Cicco, et al. The web, the users, and the mos: Influence of http/2 on user experience. In *Passive and Active Measurements*. 2017.
10. J. Brutlag, Z. Abrams, et al. Above the fold time: Measuring web page performance visually.
11. M. Butkiewicz, H. V. Madhyastha, et al. Characterizing web page complexity and its impact. *IEEE/ACM Trans. Networking*, 22(3):943, 2014.
12. P. Charonyktakis, M. Plakia, et al. On user-centric modular qoe prediction for voip based on machine-learning algorithms. *IEEE Trans. Mob. Comput.*, 2016.
13. J. Erman, V. Gopalakrishnan, et al. Towards a spdy'ier mobile web? In *ACM CoNEXT*, pages 303–314. 2013.
14. M. Fiedler, T. Hoßfeld, et al. A generic quantitative relationship between quality of experience and quality of service. *IEEE Network*, 24(2):36, 2010.
15. Q. Gao, P. Dey, et al. Perceived performance of top retail webpages in the wild: Insights from large-scale crowdsourcing of above-the-fold qoe. In *Proc. ACM Internet-QoE workshop*. 2017.
16. Google. Spdy, an experimental protocol for a faster web. https://www.chromium.org/spdy/spdy-whitepaper.
17. ITU-T. Estimating end-to-end performance in ip networks for data application., 2014.
18. C. Kelton, J. Ryoo, et al. Improving user perceived page load time using gaze. In *Proc. USENIX NSDI*. 2017.
19. A. Langley, A. Riddoch, et al. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proc. ACM SIGCOMM*. 2017.
20. Minutes of TPAC Web Performance WG meeting. https://www.w3.org/2016/09/23-webperf-minutes.html.
21. F. Qian, V. Gopalakrishnan, et al. Tm3: Flexible transport-layer multi-pipe multiplexing middlebox without head-of-line blocking. In *ACM CoNEXT*. 2015.
22. R. Schatz, T. Hoßfeld, et al. From packets to people: Quality of experience as a new measurement challenge. In *Data traffic monitoring and analysis*. Springer, 2013.
23. T. Spetebroot, S. Afra, et al. From network-level measurements to expected quality of experience: the skype use case. In *M & N workshop*. 2015.
24. M. Varvello, J. Blackburn, et al. Eyeorg: A platform for crowdsourcing web quality of experience measurements. In *Proc. ACM CoNEXT*. 2016.
25. M. Varvello, K. Schomp, et al. Is The Web HTTP/2 Yet? In *Proc. PAM*. 2016.
26. X. S. Wang, A. Balasubramanian, et al. How speedy is spdy? In *USENIX NSDI*, pages 387–399. USENIX Association, Seattle, WA, 2014.
27. X. S. Wang, A. Krishnamurthy, et al. Speeding up web page loads with shandian. In *USENIX NSDI*. 2016.
28. Web QoE dataset. https://newnet.telecom-paristech.fr/index.php/webqoe/.