Improving web QoE monitoring for encrypted network traffic through time series modeling

Nikolas Wehner, Michael Seufert, Joshua Schüler, Sarah Wassermann, Pedro Casas, Tobias Hoßfeld

Angaben zur Veröffentlichung / Publication details:

Wehner, Nikolas, Michael Seufert, Joshua Schüler, Sarah Wassermann, Pedro Casas, and Tobias Hoßfeld. 2021. “Improving web QoE monitoring for encrypted network traffic through time series modeling.” *ACM SIGMETRICS Performance Evaluation Review* 48 (4): 37–40. https://doi.org/10.1145/3466826.3466840.

Improving Web QoE Monitoring for Encrypted Network Traffic through Time Series Modeling

Nikolas Wehner
University of Würzburg

Michael Seufert
University of Würzburg

Joshua Schüler
University of Würzburg

Sarah Wassermann
AIT Austrian Institute of Technology

Pedro Casas
AIT Austrian Institute of Technology

Tobias Hossfeld
University of Würzburg

ABSTRACT

This paper addresses the problem of Quality of Experience (QoE) monitoring for web browsing. In particular, the inference of common Web QoE metrics such as Speed Index (SI) is investigated. Based on a large dataset collected with open web-measurement platforms on different device-types, a unique feature set is designed and used to estimate the RUMSI – an efficient approximation to SI, with machine-learning based regression and classification approaches. Results indicate that it is possible to estimate the RUMSI accurately, and that in particular, recurrent neural networks are highly suitable for the task, as they capture the network dynamics more precisely.

Keywords

Machine Learning, RNN, Web QoE, QoE Monitoring

1. INTRODUCTION

Nowadays, web browsing is the ubiquitous application in the Internet. This is why the performance of the Web severely influences the success of on-line services, with the potential to affect company revenues or cause Internet Service Provider (ISP) churning. The degree of annoyance or frustration with a service can be described with the concept of Quality of Experience (QoE) [1]. This concept exists also for the Web and is mostly restricted to Web page loading times. Literature proposes different metrics for modelling Web QoE. Previous QoE models used Page Load Times (PLT) [2] to approximate user satisfaction in Web browsing. By now, it has been shown that metrics like SpeedIndex (SI) [3] or Above-The-Fold (ATF) [4] capture QoE more accurately, as they concentrate only on the browser's viewport, i.e., the visible part of the page. However, the computation of the SI is an expensive process as it is based on the visual progress and thus needs to process a video capture. Therefore, additional metrics have been proposed, e.g., the Real User Monitoring Speed Index (RUMSI) [5], which efficiently approximates the SI by using resource timings.

In contrast to other services like video streaming, the complexity of Web browsing arises with the multitude of contacted servers, which are required for composing a website with all its components. This may result in several network flows which all contribute to or even determine the QoE of a website. However, as ISPs are also interested in providing high quality services to their customers, they are forced to perform Web QoE monitoring from their customers point of view. Besides the mentioned complexity of the websites themselves, the trend towards end-to-end encryption (e.g., HTTPS), makes the task of accurate QoE monitoring even more difficult. As machine learning is a powerful tool, it is deemed to provide successful solutions to the problem.

Based on a large dataset obtained with WebPageTest – the default, open-source web performance analysis tool, this work aims at accurately estimating Web QoE metrics from encrypted network traffic, in particular the RUMSI, using machine learning approaches. First, related work is discussed in Section 2. Then, the data collection process and the subsequent extracted features are presented, before the tested models for the RUMSI estimations are described in Section 3. Besides standard machine learning models such as random forest, a neural network regressor, recurrent neural network (RNN) regressors, and RNN classifiers are tested. Obtained results for both models-performance as well as feature-importance analysis are discussed in Section 4, before Section 5 concludes this work.

2. RELATED WORK

Web QoE monitoring represents a complex problem, with multiple studies in the literature. Da Hora et al. [4] conduct user studies to investigate the relationship between the proposed Web QoE metrics in the literature and the actual Web QoE. Further, they propose a more computationally efficient approximation of ATF (AATF). They show that it is possible to predict user QoE with supervised regression models like Support Vector Machines, Random Forests, and AdaBoost when using Web QoE metrics as features.

Song et al. [7] use LSTMs to predict the ATF for a website load. However, for each website a single model is learnt and only a few websites are tested in total. Thus, no general model is created. Both aforementioned approaches are also not suitable for Web QoE inference with encrypted network traffic as they rely on application-layer measurements.

Trevisan et al. [8] propose PAIN, an unsupervised learning system able to monitor the performance of websites. Based on passive flow-level and DNS measurements, PAIN provides a real-time performance analysis.

Huet et al. [9] demonstrated that supervised learning can be applied to predict the PLT and SI from encrypted packets. In particular, WebPageTest measurements to the top 500 Alexa websites are performed under different network

This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in: ACM SIGMETRICS Performance Evaluation Review 48, 4, p. 37-40. Workshop on AI in Networks (WAIN) 2020 Milan,Italy

conditions. Based on the accumulated bytes in 100ms intervals, the SI is estimated with supervised techniques such as Random Forest, XGB, and a 1D-CNN. This work is extended in [10], where several Web QoE metrics are estimated using the cumulative byte progression on network layer as input feature. They evaluate their model on different browsers and under different conditions, and outline the problem of model generalization, introduced by the highly variable and dynamic nature of the Web. Finally, in [11] we have recently investigated the same problem, extending previous work to the mobile devices scenario. These papers are the closest to this work. However, none of them investigate the suitability and advantages of recurrent neural network approaches, which is along with the evaluated feature set one of the main contributions here.

3. METHODOLOGY

3.1 Data Collection

The data was collected with a custom measurement testbed using WebpageTest (WPT). Three different devices, namely, a smartphone (Google Pixel 2 XL), a tablet (Samsung Galaxy Tab S5e), and a normal desktop computer, were used to load the top 500 Alexa websites with Chrome. All three devices were connected to the Internet through an additional computer, which served as network emulator and on which the network traces were captured as pcaps. With the WPT platform, the measurements automatically extract about 90 different KPIs and Web QoE metrics such as PLT, SpeedIndex, etc. as well as content characteristics of the visited pages. The same pages were visited ten times for each device type, using the same access network setups. In this work, approx. 60.000 runs with either no network shaping, 0.1% packet loss, 0.5% packet loss, and 1% packet loss network shaping are analyzed.

3.2 Feature Description

As websites usually perform many different requests to different servers in short periods, a window-based approach is chosen for feature generation to model the website loading process more accurately. Therefore, the network traffic is split into windows of 100ms, as this is supposed to provide a reasonable real-time temporal granularity. In addition to the features of each window, dynamic session-based features are computed, which comprise all the previous windows and where the last window contains the final session-based features. For both window- and session-based features, the bytes, the packets, the packet inter-arrival times, and the throughput are extracted and distribution statistics like mean, max, sum, etc. are computed for each metric. Assuming that the client exchanges most network traffic with a few servers only, the top three IP addresses within a window and over the course of a session are extracted. IP addresses are also included in the feature set and are represented with integers. To accommodate for multiple IP addresses originating from the same subnet, the top IP addresses are computed for varying subnet configurations. For this purpose, subnet masks of /24, /16, /8, are considered. Using only the top IP addresses would potentially help in reducing the feature set and the amount of required computation time. Again, the distribution statistics of bytes, packets, packet inter-arrival times, and throughput are computed for both window and session for each subnet. All features are computed for the upload link, for the download link, and for both directions. In addition to the presented features, the bi-directional features also contain distribution statistics of the flows and the DNS requests for each window as well as the cumulative distribution statistics of the flows, DNS requests, and established connections for the sessions.

The ground truth of the Web QoE metrics is provided by WPT. To use these metrics also for time series classification tasks, an additional metric is introduced, which maps the state of the QoE metrics to a binary variable, where 0 indicates that the website or the QoE metric, respectively, is not yet loaded and 1 indicates a full load. These binary metrics are generated in alignment with the other features for each window, so in an interval of 100ms. For this work, the evaluation concentrates on RUMSI and binary RUMSI only. The final feature set consists of 1975 network features. Note that only sessions with a RUMSI lower than 10 seconds were used, i.e., a maximum of 100 windows are generated per session.

3.3 Implemented Models

3.3.1 *Standard Regressors*

The standard ML models k-nearest neighbors (KNN), decision tree (DT), random forest (RF), and extreme gradient boost (XGB) are implemented with scikit-learn. Note that other models were tested too, but for the sake of brevity, only the results for the KNN and the tree methods are displayed as they also outperformed the other algorithms. For each model, 80% of the data are assigned to the training set and 20% of the data to the test set and the data is min-max-normalized. Further, a 5-fold grid search is performed to tune the models' hyperparameters with respect to the mean squared error (MSE). These standard regressors ignore the window-based features and take only the final session-based features into account.

3.3.2 *Deep Neural Network Regressor*

In addition to the traditional models, a neural network regressor is implemented. For training of the neural network, the data is split into a training set, validation set, and test set, where the shares of the data for the sets are 60%, 20%, and 20%. The architecture of the neural network consists of three hidden layers with 1024 neurons each followed by a dropout layer with a rate of 20%. All hidden layers use *tanh* as activation function. The output layer uses a single node for the estimated RUMSI and has no activation function. The number of nodes in the input layer correspond to the input feature dimension. As before, this regressor also relies only on the final session-based features.

3.3.3 *Recurrent Neural Network Regressor*

Next, the recurrent neural network regression approaches are described. For the recurrent architectures, two different recurrent layers are tested. In particular, the Long-Short Term Memory (LSTM) [12] and the Gated Recurrent Unit (GRU) [13] are tested as recurrent layers for the networks. Besides the internal performance differences, the architectures for the LSTM and GRU models are identical.

The regression model uses four hidden layers, where the first three hidden layer resemble the recurrent layer with 1024, 512, and 256 neurons each using the *sigmoid* activation function and where the final hidden layer resembles a dense

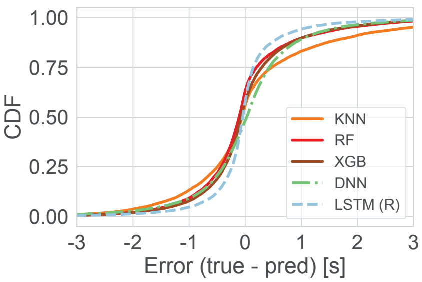

Figure 1: Model benchmarking – regression models.

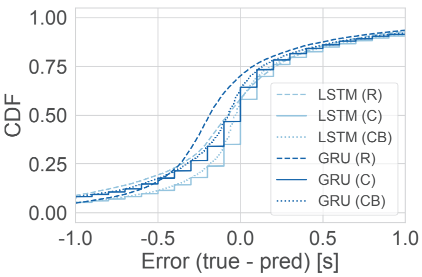

Figure 2: Model benchmarking – recurrent models.

layer with 256 neurons and again the *sigmoid* activation function. The output layer consists of a dense layer with a single unit and no activation function and corresponds to the final regressed value. The input shape corresponds to the number of slots used to create the time series, i.e., here 100 time steps, and the number of features.

In contrast to the other regression models, this model takes both the window-based features and the current session-based features per window into account.

3.3.4 *Recurrent Neural Network Classifier*

The recurrent classifier is supposed to estimate in each window whether the RUMSI has already loaded or not. This results in $|windows|$ binary predictions, which can afterwards be used to approximate the absolute RUMSI value. On one hand, this simplifies the task to learn. On the other hand, an additional bias of up to $T-1$ milliseconds is introduced, when using the start of a window as boundary and with T equal to the interval size of the window.

Again, both LSTM and GRU are tested as recurrent layers. The classification model consists of three hidden recurrent layers, which correspond to a recurrent layer with 1024, 512, and 256 neurons. The output layer consists of a dense layer with two neurons, where one neuron corresponds to not loaded (0) and the other neuron corresponds to loaded (1). These two neurons are further activated with the *softmax* function. Again, the input shape corresponds to the number of windows and the number of features. As with the recurrent regressor, both window-based features and session-based features are used as input features.

4. EVALUATION

The model-performance evaluation is conducted next, firstly by benchmarking all models together, and then by further digging deeper into the performance of the recurrent models. To shed light on the impact of the proposed input features, a feature importance analysis is also conducted.

4.1 Benchmarking

4.1.1 *All Models*

For the benchmarking, all models use the same set of features. The regression models thereby rely solely on the final session-based features.

For all deep learning models, the well-known Adam optimizer is used and the loss is calculated with the MSE for regression tasks and with the cross-entropy for the binary classification task. A batch size of 256 and a maximum of 500 epochs are used for learning. Additionally, an early stopping mechanism with a patience of 50 is implemented and the best model, i.e., the model with the lowest observed loss, is stored. The hyperparameters of all models are optimized with grid search or systematic trial and error so that the models are still comparable, e.g., with respect to the number of neurons.

Figure 1 shows the CDFs for the results of the standard supervised models, the DNN, and the LSTM regressor. The x-axis denotes the estimation error in seconds, the y-axis the fraction of website loads observing the corresponding error. It can be seen that the worst performance is shown by the KNN (orange) with a median absolute error (mAE) of 0.49, followed by the DNN (green) with a mAE of 0.41. Neglecting the LSTM (light blue), the ensemble methods RF (red, mAE: 0.35) and XGB (brown, mAE: 0.34) show the best performance as their error is most of the time below 1s. However, the LSTM regression model shows by far the best performance with respect to mAE (0.20) and MAE (0.42).

4.1.2 *Recurrent Models*

A comparison between the error CDFs of the recurrent models and approaches is presented in Figure 2. Again, the x-axis denotes the error in seconds, and the y-axis the fraction of websites.

Architectures with LSTM layers are depicted with light blue lines, while dark blue lines belong to GRU architectures. Results for the regression approaches are depicted as dashed lines (R), the classification error relative to the true RUMSI as dotted lines (CB), and the classification error relative to the specified window interval sizes (C) as solid lines. The lines for C thus show the actual prediction results of the classification approach without considering the bias described above.

It can be observed that the classification approach outperforms the regression approach for both recurrent layers significantly. While the median absolute error (mAE) for the regression approaches is around 0.29 (LSTM) and 0.30 (GRU), the mAE for the true classification error is 0.19 (LSTM) and 0.20 (GRU). Further, the introduced bias is clearly visible, but even with this bias the results for the classification approach are better.

An additional benefit of the classification approach is the fact that in the field, it is not necessary to wait until the whole page was loaded and all session-based features can be computed, Instead it is possible to predict the full load of a QoE metric in *real-time*, here with a 100ms delay. As a consequence, the QoE monitoring provides predictions and insights much faster.

With respect to the recurrent layers, it is visible that the LSTM layers offered slightly better results than the GRU

	MSE [s]		MAE [s]		mAE [s]	
	GRU	LSTM	GRU	LSTM	GRU	LSTM
A	0.82	0.87	0.52	0.51	0.25	0.20
WT	1.03	0.93	0.58	0.53	0.27	0.24
ST	0.90	0.84	0.50	0.49	0.20	0.19
W	0.94	0.96	0.55	0.54	0.26	0.23
S	1.00	0.87	0.55	0.52	0.24	0.23
TOPW	1.01	0.87	0.61	0.51	0.32	0.22
TOPS	1.00	0.90	0.57	0.53	0.26	0.25

Table 1: Feature subset analysis of classification approach.

layers. However, these results do not guarantee the superiority of LSTMs over GRUs for the suitability of Web QoE monitoring as only a small dataset is analyzed here.

Further, sessions of the three types of devices have not been distinguished. Still, it would be possible to show that a differentiation between PC and mobile devices would increase the performance as the data revealed slightly different network behavior between these device types.

4.2 Feature Importance Analysis

For the feature importance analysis, different subsets of features were trained and tested. One subset contained all features (A), another subset all windows and the corresponding top IPs (WT), and another subset all session-based features and the corresponding top IPs (ST). Further, only the windows without top IPs (W) and only the session-based features without top IPs (S) were considered. Finally, only the top IP window-based (TOPW) and the top IP session-based features (TOPS) were evaluated.

Table 1 illustrates the obtained results in terms of MSE, MAE, and mAE per recurrent layer and feature subset for the classification approach only. It is visible that the subsets including the session-based features perform mostly better than the window-based subsets. Relying only on the statistics for the top IPs also results in a low MSE, MAE and mAE. However, the best results are obtained with an LSTM and the session-based features including the top IPs.

An additional feature importance analysis with RF and XGB revealed that the features incorporating the duration of the network traffic account for the highest impact. This includes an approximation of the added up packet inter-arrival times as well as the number of observed non-empty windows.

5. CONCLUSION

In this paper, the suitability of different machine learning models for estimating the RUMSI in encrypted network traffic was investigated. A unique feature set was presented and fed into different machine learning models. It was shown that recurrent neural networks outperform traditional machine learning models and also simple neural networks with respect to several error metrics. This indicates that recurrent neural networks capture the dynamics of network traffic more efficiently. By using a window-based binary classification approach, the results could be even further improved compared to traditional regression approaches. An additional benefit of the recurrent modelling is the fact that this approach allows faster monitoring insights, as the model returns predictions for the 100ms feature windows directly, instead of having to wait for the end of the web page load to compute the required features.

The feature analysis further revealed that it is possible to already estimate the RUMSI accurately when using only the top three IP addresses and their corresponding sub-nets. Further, it could be observed that the duration of the network traffic and the packet inter-arrival times seem to be good indicators for the RUMSI estimation.

The evaluation was limited in terms of sample size. To conceive more generalizable models, a more in-depth analysis with a much larger dataset is performed in future work. In addition, only the RUMSI has been analyzed in this work. Future work includes hence also the monitoring of additional Web QoE metrics such as PLT and ATF. Finally, more sophisticated methods like recurrent neural networks with attention mechanism, temporal convolutional networks (TCN), or even recurrent graph neural networks are tested in the future to improve the monitoring performance.

6. REFERENCES

[1] P. Le Callet, S. Möller, and A. Perkis (eds), “Qualinet White Paper on Definitions of Quality of Experience,” tech. rep., European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), 2013, ver. 1.2.

[2] S. Egger, T. Hossfeld, R. Schatz, and M. Fiedler, “Waiting times in quality of experience for web based services,” in *2012 Fourth International Workshop on Quality of Multimedia Experience*, pp. 86–96, IEEE, 2012.

[3] T. Hossfeld, F. Metzger, and D. Rossi, “Speed index: Relating the industrial standard for user perceived web performance to web qoe,” in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–6, IEEE, 2018.

[4] 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 *Passive and Active Network Measurement Conference*, 2018.

[5] P. Meenan, “Real User Monitoring SpeedIndex (RUMSI),” 2020. https://github.com/WPO-Foundation/RUM-SpeedIndex

[6] E. Song, T. Pan, C. Jia, T. Huang, and Y. Liu, “Webqmon.ai: threshold-oblivious on-line web qoe assessment using lstm neural networks,” in *IEEE INFOCOM 2019 Workshops*, 2019.

[7] E. Song, T. Pan, Q. Fu, R. Zhang, C. Jia, W. Cao, and T. Huang, “Threshold-oblivious on-line web qoe assessment using neural network-based regression model,” *IET Communications*, 2020.

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

[9] A. Huet, Z. Ben Houidi, S. Cai, H. Shi, J. Xu, and D. Rossi, “Web quality of experience from encrypted packets,” in *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, pp. 30–32, 2019.

[10] 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 *Proceedings of the 2020 IFIP Networking Conference (Networking)*, 2020.

[11] S. Wassermann, P. Casas, Z. Ben Houidi, A. Huet, M. Seufert, N. Wehner, J. Schüler, S. Cai, H. Shi, J. Xu, T. Hoßfeld, and D. Rossi, “Are you on mobile or desktop? on the impact of end-user device on web qoe inference from encrypted traffic,” in *Proceedings of the 2020 IEEE International Conference on Network and Service Management (CNSM)*, 2020.

[12] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” in *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.