Investigating the Correlation between Performance Scores and Energy Consumption of Mobile Web Apps

Kwame Chan-Jong-Chu[†], Tanjina Islam[†], Miguel Morales Exposito[†], Sanjay Sheombar[†], Christian Valladares[†], Olivier Philippot[*], Eoin Martino Grua[†], Ivano Malavolta[†]
[†]Vrije Universiteit Amsterdam, The Netherlands - {k.chanjongchu | t.islam | m.e.miguel | s.sheombar | c.m.valladares}@student.vu.nl, {e.m.grua@vu.nl | i.malavolta}@vu.nl
[*]GREENSPECTOR, Nantes, France - ophilippot@greenspector.com

ABSTRACT

Context. Developers have access to tools like Google Lighthouse to assess the performance of web apps and to guide the adoption of development best practices. However, when it comes to energy consumption of mobile web apps, these tools seem to be lacking.
Goal. This study investigates on the correlation between the performance scores produced by Lighthouse and the energy consumption of mobile web apps.
Method. We design and conduct an empirical experiment where 21 real mobile web apps are (i) analyzed via the Lighthouse performance analysis tool and (ii) measured on an Android device running a software-based energy profiler. Then, we statistically assess how energy consumption correlates with the obtained performance scores and carry out an effect size estimation.
Results. We discover a statistically significant negative correlation between performance scores and the energy consumption of mobile web apps (with medium to large effect sizes), implying that an increase of the performance score tend to lead to a decrease of energy consumption.
Conclusions. We recommend developers to strive to improve the performance level of their mobile web apps, as this can also have a positive impact on their energy consumption on Android devices.

CCS CONCEPTS

• **Software and its engineering → Software performance**; • **Information systems → Web applications**.

ACM Reference Format:
Kwame Chan-Jong-Chu[†], Tanjina Islam[†], Miguel Morales Exposito[†], Sanjay Sheombar[†], Christian Valladares[†], Olivier Philippot[*], Eoin Martino Grua[†], Ivano Malavolta[†]. 2020. Investigating the Correlation between Performance Scores and Energy Consumption of Mobile Web Apps . In *Evaluation and Assessment in Software Engineering (EASE 2020), April 15–17, 2020, Trondheim, Norway.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3383219.3383239

EASE 2020, April 15–17, 2020, Trondheim, Norway

ACM ISBN 978-1-4503-7731-7/20/04…$15.00
https://doi.org/10.1145/3383219.3383239

1 INTRODUCTION

Mobile users are becoming the largest portion of consumers of Internet services [1]. By benefiting from the huge improvements of the mobile browser (*e.g.,* the HTML5 standard provides APIs for sending push notifications, geolocation, accessing the camera), a large portion of mobile users is accessing contents and services via **mobile web apps**. Mobile web apps are mobile-optimized websites accessed via the browser apps installed on users' mobile devices (*e.g.,* Google Chrome, Apple Safari, Mozilla Firefox), hosted on remote servers, and accessed via standard protocols (*e.g.,* HTTP) [2]. Mobile web apps are developed using standard programming languages such as HTML5, CSS3, and JavaScript, and a single app can run across different platforms [3], thus mitigating the well-known mobile fragmentation problem [4], both inter-platform (*e.g.,* Android vs iOS) and intra-platform (*e.g.,* being able to properly run across the plethora of Android devices available today).

Web apps that have a perceived poor *performance* can affect profits and can lead to users abandonment, specially on mobile devices where hardware and connectivity are constrained. Improving the perceived performance of (mobile) web apps is crucial for increasing conversion. For example, Pinterest rebuilt their pages for improving the performance of their web app, realizing a 40% reduction in perceived wait times, which increased both search engine traffic and sign-ups by 15% [5]. Today developers have access to tools that help them to assess the performance of their web apps and to guide the adoption of development best practices, *e.g.,* Google Lighthouse[1], WebPagetest[2], sitespeed.io[3], etc. However, unlike performance, ready-to-use tools for measuring the energy consumption of mobile web apps still have to emerge [4, 6, 7]. Without these tools, overlooking potential energy-related issues in mobile web apps can lead to the same problems lead by poor performance, such as users abandonment and low onboarding [8].

The **goal** of this paper is to investigate on the correlation between the performance scores produced by one of the most used tools for performance analysis of web apps (*i.e.,* Google Lighthouse) and the energy consumption of mobile web apps. To achieve this goal, we design and conduct an empirical experiment where *21 real mobile web apps* are analyzed in terms of both their performance level and their energy consumption. We use the Google Lighthouse analysis tool for evaluating the performance level of the mobile web apps, whereas we measure energy consumption via Greenspector,

[1]https://developers.google.com/web/tools/lighthouse
[2]https://www.webpagetest.org
[3]https://www.sitespeed.io

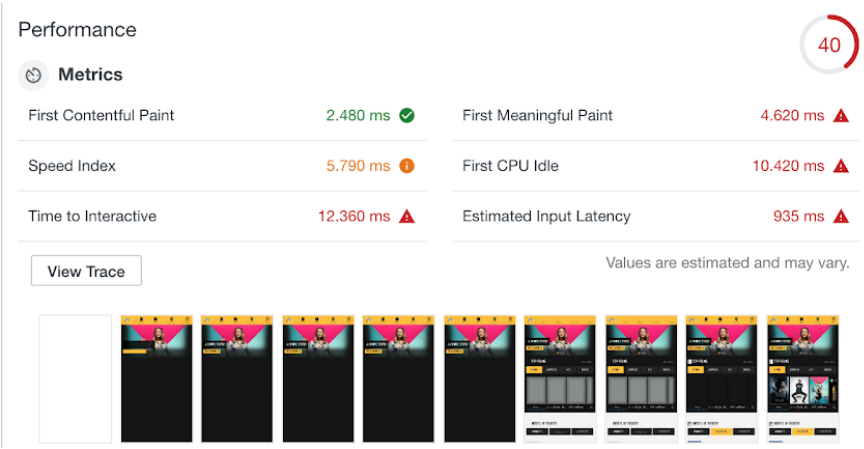

Figure 1: Web-based version of a report produced by Lighthouse

a professional software-based energy profiler for the Android platform. After having collected the measurement data for *525 runs* (21 web apps × 25 repetitions each), we statistically assess how energy consumption correlates with the obtained performance scores and carry out an effect size estimation.

The **results** of the study unveil a statistically significant negative correlation between performance scores and the energy consumption of mobile web apps (with medium to large effect sizes). This study is mainly targeting developers specialized in mobile-optimized web apps. We support them by providing evidence about the correlation between performance scores and the energy consumption of mobile web apps running on Android devices. So, specially in the early phases of a project, if a web app has a good performance score on mobile devices, then developers can use such a score as a low-cost alternative for preliminary insights also about its energy consumption.

The main **contributions** of this study are: (i) an empirical investigation on the correlation between performance scores and energy consumption of web apps running on Android devices; (ii) a discussion of the obtained results from the perspective of mobile web developers; (iii) the replication package of the study containing its results, raw data, and the scripts used to obtain the statistics.

2 STUDY DESIGN

2.1 Goal and Research Question

The goal of this study is to empirically assess to what extent the scores produced by performance audits correlate with the energy consumption of web apps running on mobile devices.

The aforementioned goal translates into the following research question, which will guide the design, conduction, analysis, and synthesis phases of the whole study: *To what extent do performance scores correlate to the energy consumption of mobile web apps?*

The main rationale behind this research question lies in the fact that today many tools exist for analyzing the performance of web apps, but there are very few ready-to-use tools supporting developers in assessing the energy consumption of their web applications. By answering this research question, we will benefit mobile developers and researchers in building an evidence-based understanding about the relationship between performance and energy consumption of mobile web apps. From a practical perspective, answering our research question can support developers since if a web app has a good performance score, developers can reasonably expect low energy consumption without needing to run dedicated, time consuming, and costly energy measurements. It is important to note that here we are not advocating to totally discard the possibility of running energy measurements with dedicated tools (which are necessary anyway), but rather we are reporting that performance analysis can be used as a low-cost alternative for getting preliminary insights about the energy consumption of the analyzed web apps, early on since the beginning of the development activities.

2.2 The Lighthouse Analysis Tool

In this study we measure the performance score of web apps via Google Lighthouse V3.0. Lighthouse is used by a large community of developers for identifying and fixing common problems emerging in web apps and it has audits for performance, accessibility, progressive web apps, web development best practices, and search engine optimization. When analyzing a target web app, Lighthouse automatically visits it multiple times and collects the metrics the developer is interested into. While running the audits, Lighthouse is able to simulate real-world conditions of mobile web apps by (i) emulating the hardware and the user agent of a mobile device and (ii) throttling the network speed, so to simulate a lossy 3G connection where packets can be lost or delayed. After running the analysis, Lighthouse produces a report with scores pertaining to each type of audits (*e.g.*, performance or accessibility) and provides actionable advice for addressing the raised issues.

For what concerns performance, Lighthouse produces metrics according to the RAIL performance model, which establishes goals based on human perception [9]. Human perception in the RAIL model relates to how an application handles four key actions: the response time to a user's input, the rendering performance for animations, optimal idle-time utilization for the sake of responsiveness, and load impact on a web app. Figure 1 shows the performance metrics produced by Lighthouse after having analysed the http://www.pathe.nl mobile web app. Specifically:

Table 1: Subjects of the study

Web app	URL	Performance score	HTML (Loc)	CSS (Loc)	JavaScript (Loc)
awsamazon	http://www.amazonaws.com	0.13 (Poor)	11642	21519	42225
apple	http://www.apple.com	0.45 (Average)	992	34967	45516
ask	http://www.ask.com	0.94 (Good)	729	557	11977
china	http://www.china.com	0.08 (Poor)	2131	3617	19791
cnn	http://www.cnn.com	0.01 (Poor)	44803	866	153037
coccoc	http://www.coccoc.com	0.22 (Poor)	1121	16149	154126
ettoday	http://www.ettoday.net	0.04 (Poor)	1599	15989	111295
hao123	http://www.hao123.com	0.17 (Poor)	14582	56	25905
instagram	http://www.instagram.com	0.69 (Average)	12078	0	83393
microsoft	http://www.microsoft.com	0.86 (Good)	3275	94	5548
paypal	http://www.paypal.com	0.63 (Average)	991	8995	20302
popads	http://www.popads.net	0.94 (Good)	434	553	7253
quora	http://www.quora.com	0.59 (Average)	4377	51492	25800
theguardian	http://www.theguardian.com	0.43 (Poor)	1337	10785	48699
tianya	http://www.tianya.cn	0.52 (Average)	347	1409	7413
twitter	http://www.twitter.com	0.48 (Average)	1616	30451	54336
whatsapp	http://www.whatsapp.com	0.63 (Average)	20532	44333	129280
xnxx	http://www.xnxx.com	0.8 (Good)	4350	10635	11023
xvideos	http://www.xvideos.com	0.76 (Good)	5381	25369	23090
yandex	http://www.yandex.ru	0.83 (Good)	20125	5054	62952
youtube	http://www.youtube.com	0.75 (Good)	25540	12318	163102

- *First Contentful Paint* represents the time when the browser renders the first bit of content [10];
- *Speed Index* represents the average time at which visible parts of the web app are displayed [10];
- *Time to interactive* measures how long a web app displays useful content (see the first metric) and event handlers are registered for most of its visible elements [10];
- *First Meaningful Paint* represents the time of the paint after which the biggest above-the-fold layout change has happened and web fonts have loaded [10];
- *First CPU Idle* represents the time when most (but potentially not all) UI elements of the web app are interactive and it is able to respond to most user input in a reasonable amount of time [10];
- *Estimated Input latency* estimates how long the web app takes to respond to user input during the busiest 5s window of page load [10].

By referring to the example in Figure 1, the analyzed web app performs well in terms of First Contentful Paint (2.4ms), whereas it has some issues with the other metrics, specially for the Estimated Input Latency (almost 1 second), meaning that in average a user can provide input to the web app only after 1 second.

All the metrics described above are combined together into a single *Performance* score, whose value can range between 0 and 100. This combined score is defined as the weighed average[4] of all the performance-related metrics ranked over the log-normal distribution derived from the performance metrics of real web apps performance data obtained from HTTPArchive [10]. Moreover, the combined Performance score is further categorized into three levels: *poor*, if the web app has a performance score between 0 and 44, *average* if its performance score is between 45 and 74, and *good* if its performance score is between 75 and 100. The thresholds of each of those levels have been defined by the Google engineers working on the Lighthouse project and guide the color coding strategy for the performance score of the tool (for example, the combined performance score in Figure 1 is shown in red because 40 is less then 45).

We decided to use Lighthouse since (i) it is open-source and steadily maintained, thus helping the replicability of the study, (i) it allows to emulate mobile devices, (iii) it can be easily integrated into our tool chain because it can be run in batch mode and via Python and Shell scripts, (iv) among others, it produces performance audits with useful metrics, and (v) the produced performance metrics are aggregated into a unique score, which captures the overall performance level of the analysed web app.

2.3 Subjects Selection

In order to be able to generalize our claims to real-world projects, we need to build a representative sample of real web apps developed in the context of real industrial projects (*i.e.*, no toy examples, no demo web apps, no web apps developed by students or non-professional developers). The starting point of our selection procedure is the Alexa Rank, *i.e.,* a dataset containing a list of the most popular websites according to Alexa Internet, a web traffic analysis company based in San Francisco and owned by Amazon[5]. The Alexa Rank provides the top 1-million websites in the world, ranked via a combination of metrics such as page views and unique site users, time-averaged over three-month periods. We choose the Alexa Rank because (i) it is well known and used in research (*e.g.,* [11, 12]), (ii)

[4]The weights are based on heuristics applied by Google engineers and are publicly available [10]

[5]http://alexa.com

its listed websites are developed by professional developers working in various companies with different technical/organizational backgrounds, and (iii) its websites are heterogeneous in terms of size, used technologies, and front-end frameworks.

Starting from the 1 million entries of the Alexa list, we programmatically select the top 100 unique web apps, discarding those web apps whose domain name is different only because of their top-level domain (*e.g.,* even if the list contains both google.com and www.google.ru, then it considers the Google web app only once). Then, we run the Lighthouse analysis in sequence on each of the 100 selected web apps via a dedicated tool called Lighthouse batch reporter [13], thus generating 100 summary reports containing all Lighthouse scores of each analyzed web app.

A preliminary analysis of the results produced by Lighthouse on the 100 web apps reveals that their performance scores are quite spread, ranging from a minimum of 0.01 (cnn) to a maximum of 1.0 (wikipedia), with a mean (median) of 0.52 (0.57). In order to ensure that our statistical analysis will consider web apps with different levels of performance, we do a *stratified random sampling* [14] across the 100 previously selected web apps. More specifically, firstly each of the 100 web apps has been classified according to the three performance levels defined by Google engineers [15]. Then, from each of these 3 performance levels (*i.e.,* poor, average, and good) we randomly select 7 web apps, yielding a total of 21 subjects. Table 1 reports the selected subjects. The table also shows that the subjects of this study are quite heterogeneous in terms of size, making us reasonably confident about the representativeness of the subjects selected for the study.

2.4 Variables and Hypothesis

The **independent variable** of this study is the *performance level* of the web app as it has been defined in Section 2.3. So, for each run of the experiment we control and change our independent variable via the selection of web apps belonging to one of the $< poor, average, good >$ performance levels.

The **dependent variable** of this study is the *energy consumption* of the web app in Joules, while it is being loaded in the mobile browser. More specifically, energy consumption is computed by (i) measuring the *power* consumed by the browser while loading the currently considered subject (in microwatts), (ii) keeping track of the *profilingTime* (in milliseconds) for each run of the experiment, defined as the difference between the timestamp of the first launch of the web app in the browser and its time to interactive, and (iii) applying the following formula for obtaining the amount of energy consumed by the web app (in Joules).

$$Energy = (\frac{power}{10^6})W \times (\frac{profilingTime}{1000})s \quad (1)$$

For what concerns our **hypotheses**, in order to answer the research question of this study we formulate the following two-tailed null hypothesis.

$$H_0 : \mu_{poor} = \mu_{average} = \mu_{good} \quad (2)$$

where μ_{poor}, $\mu_{average}$, and μ_{good} represent the mean energy consumption of the measured web apps having a *poor*, *average*, or *good* performance level, respectively. In other words, the null hypothesis states that the mean energy consumption does not significantly differ among web apps having different performance levels. Conversely, the alternative hypothesis is formulated as follows.

$$\begin{aligned} H_a : (\mu_{poor} \neq \mu_{average}) \vee \\ (\mu_{poor} \neq \mu_{good}) \vee \\ (\mu_{average} \neq \mu_{good}) \end{aligned} \quad (3)$$

Intuitively, the alternative hypothesis states that the mean of the energy consumption significantly differs among web apps having different performance levels for at least one pair of performance levels. According to the above mentioned variables and hypotheses, the experiment is designed as a 1 factor - 3 treatments experiment [14], where the factor is the performance level and the three treatments correspond to the performance levels defined in Lighthouse, *i.e., poor*, *average*, and *good*. The energy consumption variable is the outcome of the experiment. Finally, it is important to note that the experiment is balanced with respect to its factor since every treatment contains exactly 7 unique web apps, each of them belonging to the same performance level represented by the treatment.

2.5 Data Analysis

We answer our research question in four phases: exploration, check for normality and transformations, hypothesis testing, effect size estimation.

Exploration. In the first phase, we get a first indication about the obtained energy consumption values via a combination of descriptive statistics, histograms, and boxplots.

Check for normality and transformations. We analyze the distribution of the measured energy consumption across all subjects in order to check whether parametric or non-parametric statistical tests can be applied [14]. Specifically, we check if the energy consumption values are normally distributed via (i) a visual analysis of Q-Q Plots and (ii) the application of the Shapiro-Wilks statistical test [16] with $\alpha = 0.05$. We anticipate that the above mentioned analyses reveal that energy consumption is not normally distributed. As suggested in [17], we transform the energy measurement data in order to explore the possibility of having a normal distribution, which can potentially lead to higher statistical power. However, even after applying the squared, reciprocal, and log transformations, energy measurements are still not normally distributed.

Hypothesis testing. Given the results of the previous steps and the fact that the energy measurements are continuous and independent from each other, we apply the Kruskal-Wallis test (with $\alpha = 0.05$), *i.e.,* a non-parametric test for testing whether two or more samples all come from identical populations [18]. The Kruskal-Wallis test just checks if our 3 treatments come from the same populations. In order to identify which pairs of Lighthouse performance levels are significantly different, we apply the Dunn Test as post-hoc analysis [19]. Also, since we are applying multiple statistical tests, in order to reduce the chance of Type-I error we correct the obtained p-values via the Bonferroni p-value adjustment procedure [20], leading to a final α threshold equal to 0.016.

Effect size estimation. We statistically assess the magnitude of the differences between the energy consumption of web apps by applying the Cliff's Delta statistical test to each pair of performance levels [21]. The Cliff Delta is a non-parametric effect size for ordinal

variables and it does make any assumptions about the distributions being compared. The values of the obtained Cliff Delta measures are interpreted according to the guidelines proposed by [22].

2.6 Study Replicability

For more details on the research method, research execution, and extracted data, we refer the reader to the replication package of this study[6]. The package is made available with the aim of supporting independent verification and replication of this study. It contains (i) the Python scripts for performing the subjects selection, (ii) the raw data containing all the measures collected during the execution of the experiment, (iii) the R scripts for analysing the experimental data, and (iv) a detailed guide for replicating the experiment.

3 EXPERIMENT EXECUTION

To measure the energy consumption of the 21 web apps we use a power and performance profiling infrastructure called Greenspector[7]. For each device managed by Greenspector, a calibration process is run to give a probe trust level. This level takes into account the stability of the measure and the frequency. In this study we use an HTC Nexus 9 tablet running Android 7.1.1. The device is equipped with a 2.3 GHZ Dual Core processor (Denver), 2Gb of RAM, a Kepler DX1 GPU and a 802.11 a/b/g/n/ac Wi-Fi interface. Furthermore, the device has a 6700 mAh Lithium-Polymer battery. For the Nexus 9, the trust level is 8 on 10 (Calibration on a small range and precise measure). To get a better indication of the energy consumed by each web app, we consider the reference power consumption value, provided by Greenspector, which removes the battery drainage caused by the Android operating system. The reference power consumption is obtained by measuring the system with a browser opened on a blank screen.

For orchestrating the execution of all the runs of the experiment we make use of Android Tests developed in UIAUtomator[8]. Greenspector is integrated as an API in these tests. Tests are run with Greenspector Testrunner which permits to communicate using Android Debug Bridge (ADB[9]) over Wi-Fi.

In our experiment, Testrunner is executed on a laptop with Linux Mint 19 Cinnamon 3.8.9, Intel i5-5200U and 16GB RAM. Web apps are run within the Google Chrome browser (version 54.0.2840.85). With the laptop connected to the Nexus 9, all the web apps are automatically loaded by the test in the Google Chrome app running on the device, while their energy consumption is measured via Greenspector. Both the laptop and the Nexus 9 run under the same Wi-fi network with a speed of 100 Mbps. To ensure that the Wi-Fi conditions do not alter the results of the experiment, the Nexus 9 and the laptop are always placed 5 meters from the Wi-Fi router. Further, we take special care in keeping the execution environment as clean as possible, Greenspector Testrunner permit to manage this environment, specifically: the Nexus 9 is loaded with a clean installation of the Android OS, it has been configured so to do not perform any OS updates, Google services have been disabled, all third-party apps have been uninstalled, a whitelist permit to only autorize minimum applications, and push notifications have been disabled as well. In order to take into account the intrinsic variability of energy measurement, we take the following precautions: (i) the measurement of each web app is repeated 25 times, (ii) between each run the Nexus 9 remains idle for 2 minutes so to take into account tail energy usage, i.e., the phenomenon where where certain hardware components of mobile devices are optimistically kept active by the OS to avoid startup energy costs [25], and (iii) the Google Chrome app is cleared before each run so to reset its cache and persisted data.

[6]https://github.com/S2-group/ease-2020-replication-package
[7]https://greenspector.com/en/
[8]https://developer.android.com/training/testing/ui-automator
[9]https://developer.android.com/studio/command-line/adb

4 RESULTS

In this section we report the results we obtained in each of the four phases of our data analysis: exploration, check for normality and transformations, hypothesis testing, and effect size estimation.

4.1 Data Exploration

The energy consumption across all 21 web apps ranges between 7.9 Joules and 210.46 Joules, with a mean (median) of 54.51 (40.99) Joules a standard deviation of 43.84 Joules.

Table 2: Subjects of the study

Web app	Duration (s)	Platform CPU (%)	Mean energy consumption (J)
awsamazon	26.95945	70.43	193.97
apple	6.92678	62,8	37.67
ask	3.36417	54.94	13.79
china	14.12801	56.90	64.83
cnn	32.48561	53.57	143.07
coccoc	14.36013	49.21	56.07
ettoday	16.07516	65.96	83.96
hao123	12.86212	51.62	54.46
instagram	5.60932	65.71	32.03
microsoft	3.77544	80.11	24.75
paypal	5.06258	60.83	25.30
popads	2.75625	68.08	14.60
quora	7.74541	62.35	45.43
theguardian	13.14062	76.96	98.17
tianya	5.15653	34.87	11.83
twitter	7.37284	72.92	48.28
whatsapp	7.25573	69.67	40.89
xnxx	5.83986	74,22	37.43
xvideos	6.61091	85.75	51.72
yandex	5.07072	79.52	34.91
youtube	6.09105	61.43	30.77

Table 2 shows a breakdown of the mean energy consumption of each web app. Here it is evident that awsamazon and cnn are consuming a much higher amount of energy with respect to the other subject web apps. We can motivate this observation by the fact that those two web apps have the highest time to interactive with respect to all the others (about 27s and 32s respectively, as shown in Table 2). It is also interesting to note that those two web apps have the lowest performance score (0.13 and 0.01, respectively), providing

us a first informal indication about the relationship between the performance score, time to interactive, and energy consumption.

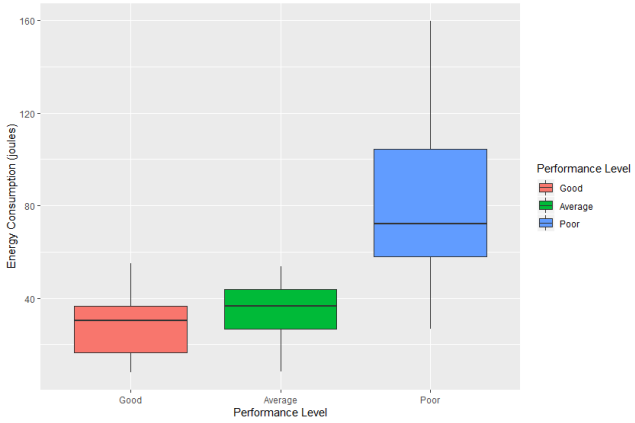

Figure 2: Energy consumption per treatment

To better understand the behavior of the energy distribution with respect to the performance analysis produced by Lighthouse, in Figures 2 and 3 we zoom-in on the energy consumption per performance level. From the diagrams, we observe that the energy consumption measurements are very similar for both *good* and *average*, whereas a larger difference is observed when comparing these results to energy consumed by web apps belonging to the *poor* performance level. Moreover, here we can also observe that the energy consumption values are much more spread when considering web apps belonging to the *poor* performance level with respect to the ones belonging to the other levels.

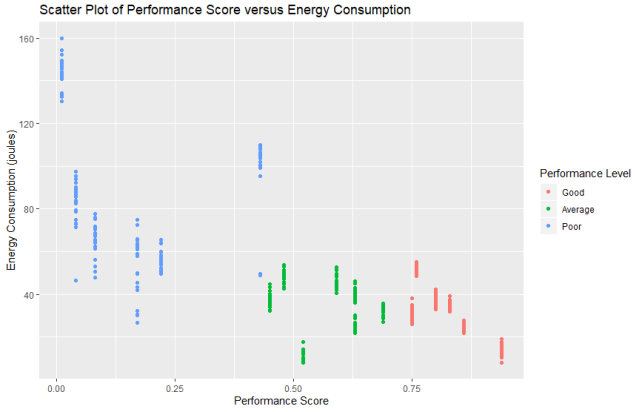

Figure 3: Performance scores vs energy consumption

The exploration of the obtained data makes us suspect that web apps with *poor* performance tend to also consume more energy consumption and with higher variability; this phenomenon may be explained by the longer execution times of the web apps having *poor* performance, specially in the cases of awsamazon and cnn.

4.2 Check for normality and transformations

Figure 4 shows the distribution of the density of the energy consumption across all 21 web apps. Visually this data does not look normally distributed. For further proof, we produce a Q-Q plot of the measured energy consumption against a random sample taken from a normal distribution (see Figure 5). Also the Q-Q plot confirms that the collected data about energy consumption is not normally distributed.

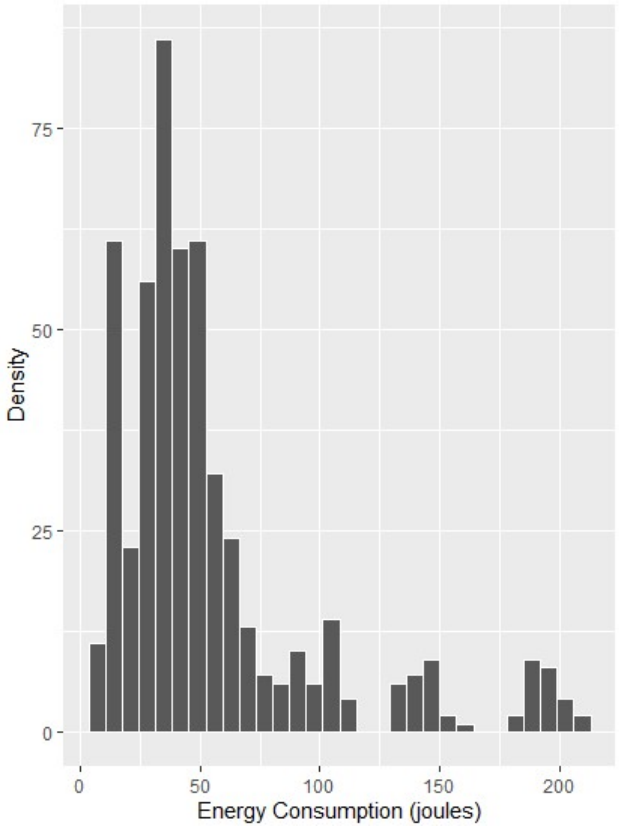

Figure 4: Distribution of the density of the energy consumption

Having normally distributed data may allow us to apply parametric statistical tests, thus leading to higher statistical power, we transform the energy measurement data by applying the squared, reciprocal, and log transformations [17].

In Figures 6 and 7 we observe that both the square and the reciprocal transformations exacerbate the already-skewed nature of the data. Indeed, both histograms are not exhibiting the characteristic bell-shape curve of a normal distribution and the Q-Q plots do not provide any indication of having normally distributed data.

When applying the log transformation (see Figure 8), the shape of the data seems to be close to that of a normally distributed one, however the Q-Q plot still shows a slight curvature. To get a further indication, we performe the Shapiro-Wilk test [16] which results in a p-value of $4.4 * 10^{-7}$, thus rejecting the null hypothesis that the transformed data lies within a normal distribution.

In conclusion, the energy consumption we measured during the experiment execution is not normally distributed, even when applying various data transformation operations.

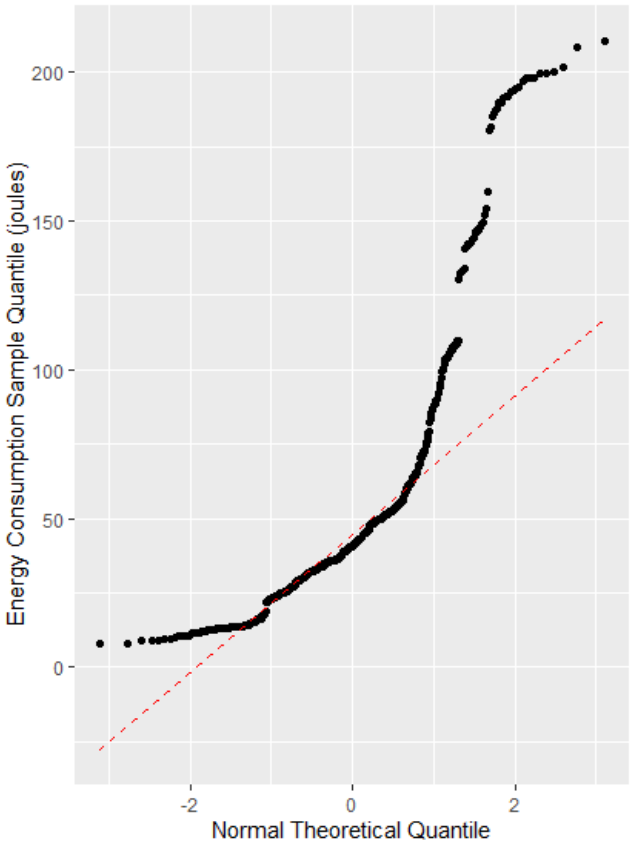

Figure 5: Q-Q-plot of the energy consumption

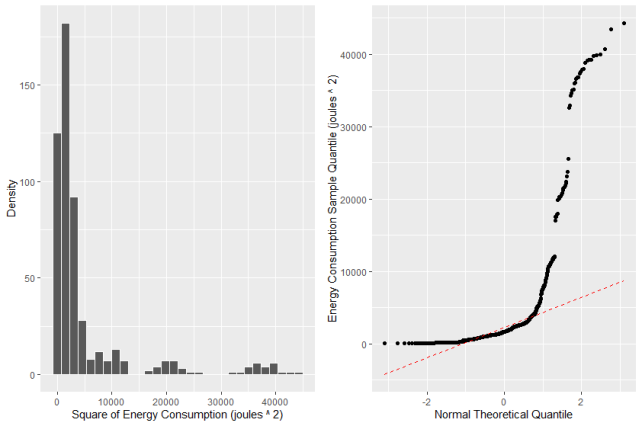

Figure 6: Squared transformation for energy consumption

4.3 Hypothesis testing

The application of the Kruskal Wallis test produces a p-value of $2.2{*}10^{-16}$, therefore allowing us to *reject* the null hypothesis stating that the energy consumption measures at each performance level come from identical populations. This result provides evidence that

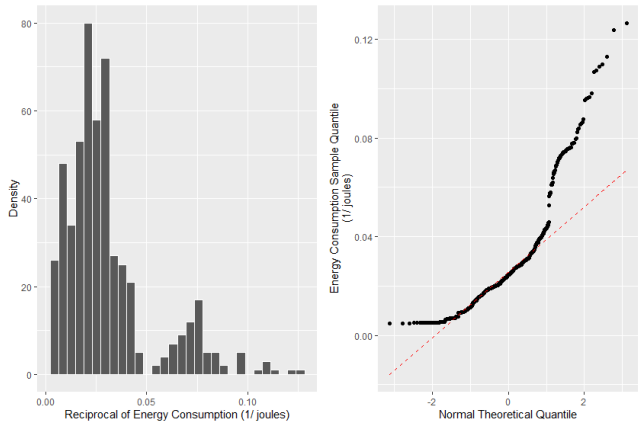

Figure 7: Reciprocal transformation for energy consumption

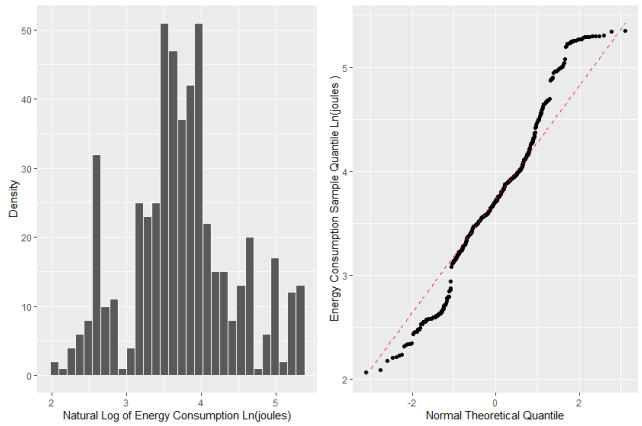

Figure 8: Log transformation for energy consumption

web apps belonging to different performance levels have different energy consumptions.

Following, we apply the Spearman's rank correlation test [23] to asses the correlation between the performance score and energy consumption of the web apps. We obtain a negative correlation value of -0.69 for energy consumption w.r.t. performance, which infers that the energy consumption of the considered web apps increases as the performance score drops.

In light of these results we perform a pairwise comparison using the Dunn test [19] (with Bonferroni correction). The resulting p-value for the *good-poor* pair is $3.70{*}10^{-56}$, whereas the *average-poor* pair obtained a p-value of $8.39 * 10^{-40}$. The same can be concluded when considering the *good-average* pair, as the resulting p-value is $2.23 * 10^{-2}$. These results show clear significance between the three pairs, therefore providing evidence about the fact that web apps belonging to different performance levels (as indicated by Lighthouse) consume a significantly different amount of energy, when running in the Google Chrome browser in Android.

4.4 Effect size estimation

As a follow-up to our tests, we investigate on the effect size in the context of the results of our Dunn tests. To do so, we apply the Cliff's delta measure [21]. A *large* effect size, *i.e.,*-0.95 and -0.94, is found when considering the *good-poor* and the *average-poor* pairs. Differently, a *small* effect (*i.e.,*-0.22) size is obtained when considering the *good-average* pair.

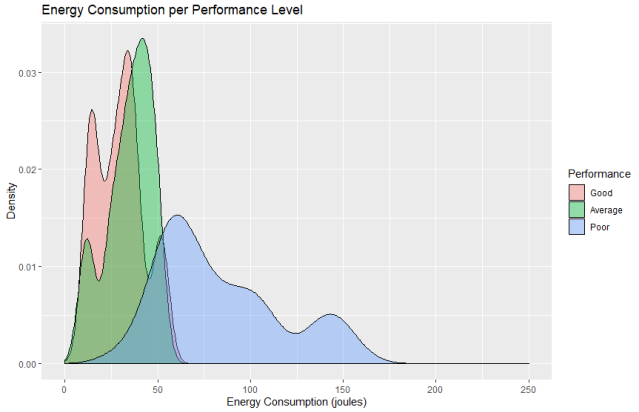

Figure 9: Density curve for energy consumption

To further support the results found with the application of the Cliff's delta measure, in Figure 9 we show how the density of the values of energy consumption varies among the Lighthouse performance levels. We observe that the differences between the *poor* performance level and the other two levels is indeed large, whereas the differences between the *average* and *good* levels is much smaller (but still present).

5 DISCUSSION

From the results of the statistical tests performed in the previous section we can elaborate on our research question. The outcome of the Kruskal Wallis test shows that we can reject our null hypothesis. Therefore, we can say that there is a statistically significant difference in the energy consumption of web apps belonging to different performance levels.

The application of the Dunn's test shows that there is a significant difference also between pairs of performance levels (*i.e.,* good-average, good-poor, average-poor). Moreover, the test shows that the difference in energy consumption of web apps with poor performance levels is quite relevant and the difference between good and average is also quite distinctive.

Finally, to evaluate the degree of overlap between two distributions of performance levels, we apply the Cliff's delta effect size measure. This yields a large effect size between good-poor and average-poor performance levels, while it results in a medium effect size for good-average web apps, which is still a significant value and therefore cannot be negligible. Looking into the differences in effect size, we can safely deduce that there are diminishing effects as web apps become better in terms of performance.

The performance scores produced by Google Lighthouse can be used by developers as an indication of how much energy a certain web app may consume. Moreover, according to our experiment setup, developers can achieve large gains in terms of energy consumption when improving the performance of their web apps from a *poor* to an *average-good* level, while a medium gain can be achieved when improving the performance of the web apps from an *average* to a *good* level. Nevertheless, according to our empirical results, performance analysis can be used as a low-cost alternative for getting preliminary insights about the energy consumption of the mobile web apps running in Google Chrome on Android, specially as a low-cost alternative for preliminary insights about their energy consumption. In any case, if the developer requires precise and/or advanced analyses of the energy consumption of their web apps, going with a proper measurement infrastructure and/or using professional services is still needed. It is also important to note that the relationship between performance and energy is not yet fully explored and in some cases better performance scores may not certainly lead to lower energy consumption (*e.g.,* faster web apps may consume more CPU power). Again, if the developer needs a deep assessment of the energy footprint of their web apps, setting up a dedicated measurement infrastructure is advised.

6 THREATS TO VALIDITY

6.1 External Validity

The population of the subject web apps was chosen from the Alexa top 1 million web apps in terms of traffic. From that, we randomly selected 21 out of the 100 most-visited web apps as our representative sample. We chose to make the selection from such category in order to (i) be sure that the considered subjects are real-world web apps developed by professional developers and (ii) guarantee that the selected web apps would be of interest to a general population. The randomization of the selection process allows us to not introduce biases based on the type of web apps selected.

Furthermore, we utilized a relatively modern device (*i.e.,* an HTC Nexus 9) with relatively good hardware specifications. Google Chrome was the mobile browser of choice as it captures 61.77% of the market share on mobile devices [10]. With this choice of tools we can be reasonably sure to have a realistic experimental setting that can directly translate into a real world scenario. Nevertheless, newer devices running newer Android releases may lead to different energy measurements; further replications of the performed experiments can help in mitigating this potential threat to validity.

6.2 Internal Validity

Maturation might play a role if a trial is repeated multiple times over same object. In order to mitigate this potential threat to validity, we made sure that each run of the experiment is executed at intervals of 2 minutes. After each execution, we also clear up the cache of the web browser.

Moreover, there are several factors that can affect the reliability of the measures i.e. brightness of the device, distance to the router and interference with other processes consuming energy. We aimed at mitigating those potential threats to validity as much as possible by setting up a minimal and replicable measurement infrastructure (see Section 3). Finally, in this study we are collecting

[10] http://gs.statcounter.com/browser-market-share

energy measures by using a software-based power profiler (*i.e.,* Greenspector); this potential source of bias is partially mitigated by the device-specific calibration step of Greenspector. Future replications of the experiment using different (hardware) measurement infrastructures can further mitigate this potential threat to internal validity.

6.3 Construct Validity

In order to mitigate potential inadequate per-operational explanation of constructs, we defined *a priori* all the details related to the design of the experiment (*e.g.,* the goal, research question, variables, data analysis procedures), before we started any experiment run. Our experiment is based a single factor (*i.e.,* the performance level produced by Lighthouse), which is the independent variable of our experiment design. This may potentially lead to mono-operation and mono-method biases. We mitigated those potential biases by (i) considering 21 different subjects in the experiment, (ii) selecting the subjects via a stratified strategy (7 web apps for each Lighthouse performance level), and (iii) we executed 25 repetitions for each subject.

6.4 Conclusion Validity

This kind of threats is about the relationship between treatment and outcomes of the study. In this study, statistical analysis assumptions have been accounted to minimize the possibility of false results being reported. Moreover, the majority of the statistical tests produced p-values far below the 0.05 significance level. To minimize the error rate of the results, the Bonferroni correction was adopted to adjust the significance level when applying the Dunn test. Finally, a complete replication package is publicly available for independent verification and inspection of each step of the performed experiment.

7 RELATED WORK

The literature provides a series of studies targeting either performance or energy efficiency in the context of **mobile web apps**. For example, Thiagarajan *et al.* performed an in-depth analysis of the energy consumption of mobile web browsers [8]. The study involved the measurement of the energy consumption of 26 web apps, which have been performed using a hardware multimeter and a patched version of the Android browser. This experimental setup allowed them to measure also the energy needed to render individual web elements (*e.g.,* images, JavaScript files). Based on the obtained results, the authors provided guidelines for building more energy-efficient web pages without affecting the user experience.

Malavolta *et al.* performed an empirical study on the energy efficiency of service workers in the context of progressive web apps [24]. The experiment considered the impact of the use of service workers and the type of network available (2G or WiFi) on the overall energy consumption of 7 progressive web apps running on two different Android devices. Their main result is that service workers do not have a significant impact over energy consumption, regardless of the network conditions.

Nejati and Balasubramanian carried out an empirical study about the performance bottlenecks of mobile and Desktop browsers [25]. At the core of their measurement infrastructure lies WProf-M, an extension of the Android Chromium browser which includes a software-based performance profiler. Then, they run the experiment using both web apps belonging to a dedicated experimental testbed and a set of 200 real-world web apps mined from the Alexa list. Their results provide evidence about the fact that computation activities (instead of networking ones) are the main bottleneck in mobile browsers; in Desktop browsers the result is totally different, where networking activities are the main bottleneck.

Vesuna *et al.* carried out a study on how caching impacts the mobile browser performance [26]. In that study, a dataset of 400 web pages has been used. Among the various results, Vesuna *et al.* discovered that in case of a perfect cache hit ratio, the reduction of page load times on mobile devices is much lower than that on Desktop setups.

All the above mentioned studies share some methodological and technical aspects with ours (*e.g.,* parts of the experiment design), but they all differ in the goal of the experiment since they focus exclusively either on performance or energy consumption of web apps, whereas we focus on better characterizing the relationship between them and how already existing performance analysis tools like Google Lighthouse can be used as a proxy for energy assessment.

Another interesting line of research related to this study is about empirical studies where **native mobile apps** are measured for better understanding their characteristics in terms of performance and energy efficiency.

Cruz and Abreu performed an investigation on whether fixing performance-related code smells in native Android apps also leads to an improvement in terms of energy consumption [27]. Specifically, they mined 6 apps from the FDroid repository of open-source Android apps and identified recurrent performance-related code smells via the well-know Android Lint static analysis tool. Then, for each detected smell, the app was manually refactored and a new version of the app was produced. Finally, a set of automated UI test cases was developed and executed on the apps while they were running on a bare-board computer ODROID-XU. The results of the experiment revealed that performance-related code smells actually lead to more energy-efficient mobile apps, saving up to an hour of battery life. Even though it focusses on native Android apps, this result is in line with the findings of our study and confirms that in Android performance measurement can be seen as a good proxy for energy consumption.

Similarly to the previous study, Palomba *et al.* investigated on the impact of code smells on the energy consumption of native Android applications [28]. This study has a larger scale and involves a set of 60 real Android apps belonging to the dataset built by Choudhary *et al.* [29]. For the detection of code smells they rely on aDoctor, a code smell detector developed by the same research group and able to extract structural properties from the source code of Android apps to detect instances of code smells like durable wakelock, leaking thread, slow loop [30]. Energy profiling is based on Petra, a software-based tool that is able to estimate the energy consumed by Android apps at the method level [31]. The results of this study unveil that the some code smells have a strong impact on the energy efficiency of source code methods. Moreover, it emerged that refactoring code smells is a key activity to improve energy efficiency of Android apps.

Gottschalk *et al.* investigated the energy impact of refactoring 5 code smells from 2 Android apps [32]. Identified smells were related to mobile energy consumption, *i.e.*, targeting 3rd-party advertising and management of specific mobile resources (Wifi, GPS, display). The impact evaluation of the smell shows that most refactorings seem to reduce energy consumption, albeit with varying figures (from 5 to 30%).

Finally, Rodriguez *et al.* [33] investigate the energy impact of refactoring 2 smells (namely, God Class and Brain Method) from 3 Android software applications. Authors conclude there is a trade-off between quality of design (hence amount of refactored code smells) and energy efficiency.

Differently from the studies above, our study targets web apps running on standard mobile browsers, instead of focussing on native apps directly running on the Android OS. Involved technologies are totally different, thus potentially leading to totally different results. Interestingly, both our study and the mentioned ones confirm the tight relationship between performance and energy consumption in the Android platform. Also, from both our study and the other studies on native apps it emerged that the relationship between performance and energy consumption is not predictable in all the cases, calling for further studies in this research area.

8 CONCLUSIONS

In this article we study the relation between the performance score given by the Lighthouse analysis tool and the energy consumption of mobile web apps. To do so, we designed a controlled experiment involving 21 real third-party web apps. Our results show a significant negative correlation between the performance score and the energy consumption of a mobile web app. Overall, we can state that the performance analysis tools like Lighthouse can be used as proxies for energy consumption. In addition to provide performance score, Lighthouse provides also guidance on how to improve critical performance aspects of web apps. We therefore recommend developers to use Lighthouse development milestones in order to improve the web app's performance. In doing so, developers will increase also the likelihood of lowering the energy consumption of their web apps, as showed by this study.

For future work, our study will be extended by performing an analysis including each of the audits used to give the aggregated performance score in Lighthouse. This would make it possible to determine which audit affects more the energy consumption. Moreover, we are planning to replicate the experiment by using additional performance analysis tools and metrics so to improve the generalizability of our findings.

REFERENCES

[1] Statscounter, "Desktop vs mobile vs tablet market share worldwide," GlobalStats, Tech. Rep., August 2018. [Online]. Available: http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet

[2] I. Malavolta, "Beyond native apps: web technologies to the rescue!(keynote)," in *Proceedings of the 1st International Workshop on Mobile Development*. ACM, 2016, pp. 1–2.

[3] "Native, Web or Hybrid Mobile-app Development," *White paper, IBM Corporation*, April 2012, document Number: WSW14182USEN.

[4] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real challenges in mobile app development," in *International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2013, pp. 15–24.

[5] V. A. . J. C. Sam Meder. (2017, March) Driving user growth with performance improvements. [Online]. Available: https://medium.com/@Pinterest_Engineering/driving-user-growth-with-performance-improvements-cfc50dafadd7

[6] G. Pinto and F. Castor, "Energy efficiency: A new concern for application software developers," ACM, Tech. Rep., 2018. [Online]. Available: http://gustavopinto.org/lost+found/cacm2017.pdf

[7] M. Nagappan and E. Shihab, "Future trends in software engineering research for mobile apps." in *FOSE@ SANER*, 2016, pp. 21–32.

[8] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh, "Who killed my battery?: analyzing mobile browser energy consumption," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 41–50.

[9] M. K. . A. O. . K. B. . J. Miller. (2018, August) Measure performance with the rail model. [Online]. Available: https://developers.google.com/web/fundamentals/performance/rail

[10] Google. (2019, February) Lighthouse scoring documentation. [Online]. Available: https://github.com/GoogleChrome/lighthouse/blob/master/docs/scoring.md

[11] S. Mahajan, N. Abolhassani, P. McMinn, and W. G. Halfond, "Automated repair of mobile friendly problems in web pages," in *Proceedings of the 40th International Conference on Software Engineering*. ACM, 2018, pp. 140–150.

[12] F. S. Ocariza Jr, K. Pattabiraman, and B. Zorn, "Javascript errors in the wild: An empirical study," in *22nd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2011, pp. 100–109.

[13] (2018, october) Lighthouse batch reporter. [Online]. Available: https://www.npmjs.com/package/lighthouse-batch

[14] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[15] Google. (2018) Lighthouse v3 scoring guide. [Online]. Available: https://developers.google.com/web/tools/lighthouse/v3/scoring#perf-scoring

[16] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[17] S. Vegas, "Analyzing software engineering experiments: everything you always wanted to know but were afraid to ask," in *Proceedings of the 39th International Conference on Software Engineering Companion*. IEEE Press, 2017, pp. 513–514.

[18] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.

[19] O. J. Dunn, "Multiple comparisons among means," *Journal of the American statistical association*, vol. 56, no. 293, pp. 52–64, 1961.

[20] C. Bonferroni, "Teoria statistica delle classi e calcolo delle probabilita," *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, vol. 8, pp. 3–62, 1936.

[21] N. Cliff, "Dominance statistics: Ordinal analyses to answer ordinal questions." *Psychological bulletin*, vol. 114, no. 3, p. 494, 1993.

[22] R. J. Grissom and J. J. Kim, *Effect sizes for research: A broad practical approach*. Lawrence Erlbaum Associates Publishers, 2005.

[23] *Spearman Rank Correlation Coefficient*. New York, NY: Springer New York, 2008, pp. 502–505. [Online]. Available: https://doi.org/10.1007/978-0-387-32833-1_379

[24] I. Malavolta, G. Procaccianti, P. Noorland, and P. Vukmirović, "Assessing the impact of service workers on the energy efficiency of progressive web apps," in *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems*. IEEE Press, 2017, pp. 35–45.

[25] J. Nejati and A. Balasubramanian, "An in-depth study of mobile browser performance," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 1305–1315.

[26] J. Vesuna, C. Scott, M. Buettner, M. Piatek, A. Krishnamurthy, and S. Shenker, "Caching Doesn't Improve Mobile Web Performance (Much)," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. USENIX Association, 2016.

[27] L. Cruz and R. Abreu, "Performance-based guidelines for energy efficient mobile applications," in *Mobile Software Engineering and Systems (MOBILESoft), 2017 IEEE/ACM 4th International Conference on*. IEEE, 2017, pp. 46–57.

[28] F. Palomba, D. Di Nucci, A. Panichella, A. Zaidman, and A. De Lucia, "On the impact of code smells on the energy consumption of mobile applications," *Information and Software Technology*, vol. 105, pp. 43–55, 2019.

[29] S. R. Choudhary, A. Gorla, and A. Orso, "Automated test input generation for android: Are we there yet?(e)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 429–440.

[30] F. Palomba, D. Di Nucci, A. Panichella, A. Zaidman, and A. De Lucia, "Lightweight detection of android-specific code smells: The adoctor project," in *24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2017, pp. 487–491.

[31] D. Di Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, and A. De Lucia, "Petra: a software-based tool for estimating the energy profile of android applications," in *Proceedings of the 39th International Conference on Software Engineering Companion*. IEEE, 2017, pp. 3–6.

[32] M. Gottschalk, J. Jelschen, and A. Winter, "Saving energy on mobile devices by refactoring," in *EnviroInfo*, 2014, pp. 437–444.

[33] A. Rodriguez, M. Longo, and A. Zunino, "Using bad smell-driven code refactorings in mobile applications to reduce battery usage," *Simposio Argentino de*, 2015.