information

MDPI

Article

Towards Integrating Mobile Devices into Dew Computing: A Model for Hour-Wise Prediction of Energy Availability

Mathias Longo [1], Matías Hirsch [2,*], Cristian Mateos [2] and Alejandro Zunino [2]

1 Viterbi School of Engineering, University of Southern California, Los Angeles, CA 90007, USA; mathiasl@usc.edu
2 ISISTAN-UNICEN-CONICET, Tandil B7001BBO, Buenos Aires, Argentina; cristian.mateos@isistan.unicen.edu.ar (C.M.); alejandro.zunino@isistan.unicen.edu.ar (A.Z.)
* Correspondence: matias.hirsch@isistan.unicen.edu.ar; Tel.: +54-249-438-5650 (ext. 2303)

Received: 17 December 2018; Accepted: 21 February 2019; Published: 26 February 2019

Abstract: With self-provisioning of resources as premise, dew computing aims at providing computing services by minimizing the dependency over existing internetwork back-haul. Mobile devices have a huge potential to contribute to this emerging paradigm, not only due to their proximity to the end user, ever growing computing/storage features and pervasiveness, but also due to their capability to render services for several hours, even days, without being plugged to the electricity grid. Nonetheless, misusing the energy of their batteries can discourage owners to offer devices as resource providers in dew computing environments. Arguably, having accurate estimations of remaining battery would help to take better advantage of a device's computing capabilities. In this paper, we propose a model to estimate mobile devices battery availability by inspecting traces of real mobile device owner's activity and relevant device state variables. The model includes a feature extraction approach to obtain representative features/variables, and a prediction approach, based on regression models and machine learning classifiers. On average, the accuracy of our approach, measured with the mean squared error metric, overpasses the one obtained by a related work. Prediction experiments at five hours ahead are performed over activity logs of 23 mobile users across several months.

Keywords: mobile cloud computing; dew computing; battery prediction; feature selection; machine learning

1. Introduction

Dew computing is conceived as the bottom structural layer of the existing distributed computing hierarchy [1] integrated by the so called cloud-edge/fog computing services. In such a hierarchy, the most powerful and expensive storage and computation resources reside in cloud data centers. Hence, cloud services are worldwide accessible, in contrast to services offered in the next layer represented by edge/fog computing infrastructure. The latter is expected to provide services for satisfying storage and computing needs within a metropolitan area. In the last decade, many research efforts has been conducted in relation to edge/fog computing paradigm [2–4]. The high latency internetwork communication that involves using distant cloud services imposes a limitation for running applications with delay-sensitive processing requirements such as real-time analytics. In response to such limitation edge/fog computing solutions emerged and bring computing and storing resources closer to where computing needs are generated [5]. In this way, long communication latencies are mitigated by reducing the usage of resources from distant cloud data centers. However, when edge/fog resources are overloaded, i.e., when their capability is overpassed to handle peaks of high demand applications, the layer relies on its connection with the cloud resources to offload tasks. When this happens, the edge/fog layer exposes its dependency on internetwork back-haul. As a complementary bottom layer of such hierarchical resources organization is the dew layer, which aims at enhancing user experience by exploiting resources near to the end-user with minimum internet access [6]. Intuitively, resources in this layer are more limited, though accessible with lower latency than those of the above layers, and users scope is within a local area network or even a user's local machine. A key aspect to provide dew layer services is to have dew devices capable of satisfying, at least in a degraded mode, user requirements and complete autonomy to decide when to use services provided by above layers [7]. Figure 1 illustrates the users scope, resources and communication features of such complementary distributed computing paradigms.

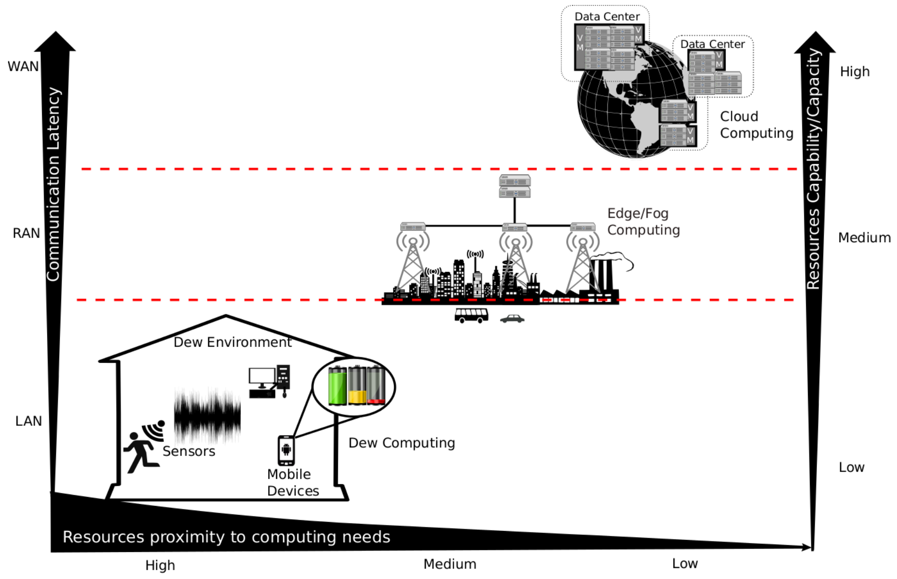

Figure 1. Cloud-edge-dew distributed computing hierarchy.

In its beginnings, dew computing related to maintaining copies of websites and databases (dew sites) in user's local computers (dew servers) to support an internet-free web surfing experience [8,9]. Dew servers collaborate with the cloud for performing tasks such as synchronization, restoring and personalization of dew sites content located in dew computing devices. These last are represented by human-operated computers like desktop computers, laptops and mobile devices including tablets and smartphones [6]. Recent conceptualizations of dew computing envisage an enriched functionality for the layer that goes beyond exploiting the storing/communicating capabilities of dew devices. In [6], the importance of an active cooperation among nearby dew devices for providing tasks execution services is highlighted. In [10], cooperation of such kinds are proposed for IoT data streams processing. The personal smart traffic lights prototype presented in [11] is a concrete example of application in the intelligent transportation domain. Moving towards processing data by prioritizing the exploitation of on-device/local network resources aims to improve the energy efficiency, trust, security, resilience to network outages, and response time properties of application execution approaches that primarily rely on remote cloud services [12].

Dew computing area implementations have unlimited applicability in indoor/outdoor ordinary-life scenarios, including home automation, smart gardening, green computation, smart health care, to mention a few examples. The integration of mobile devices as first-class computing providers of dew environments sounds like a rule more than the exceptional case for several reasons: (1) these are one of the most common computational device in the world. Their number increases year by year to the point that by 2021 it is estimated that there will be 1.5 mobile-connected devices per capita (http://tinyurl.com/mokcut3). (2) Their computing, storage and sensing capabilities are enhanced with every new generation. Smartphones are nowadays capable of running resource intensive routines, e.g., for processing medical images [13] or performing complex engineering equations [14]. (3) They share a close proximity to the end user. (4) They are able to continuously operate for several hours, days, even weeks unplugged from the electricity grid, i.e., only with the power of their Li-ion batteries.

In line with the described potential, there are many efforts which advocate making mobile devices cooperate in task execution [15]. To accomplish this, it is necessary to design task scheduling mechanisms, i.e., the logic by which tasks or data are distributed among participant battery-driven mobile nodes to be executed or processed, respectively. Task scheduling design targeting resource scavenging of a group of nearby mobile devices encloses high complexity due to the highly dynamic and heterogeneous nature of resource availability [15,16]. In other words, resource quantification used in tasks scheduling mechanisms is difficult to tackle effectively; mobile devices can change their physical location, they are non-dedicated by nature (dew tasks and device owner's applications must compete for the same resources) and have energy limitations due to the fact that mobile devices are battery-powered. Particularly, quantifying resources of a device based on its future energy availability is of utmost importance, not only to avoid incomplete task assignments caused by run out of battery events, but also for encouraging dew device owners to participate in dew computing considering at the same time the battery requirements of his/her future interactions.

In this context, we propose a model to quantify future energy availability in mobile devices by taking into account the device owner usage profile as input, which is an open problem seldom explored in the literature [17]. To limit the scope of our research, we concentrate on producing a battery prediction model suitable for prospective designs of practical energy-aware schedulers which aim at scavenging mobile devices computing capabilities.

Basically, our proposed model utilizes information of past owner's activity (charging state, screen state and brightness level, application execution, activated radios, etc.) to produce ahead predictions of remaining battery within a time window of several hours. For the model construction and the assessment of its predictive accuracy, we employed real traces of mobile device activity from the device analyzer data-set [18]. At present, this is the largest fine-grained mobile device usage data-set including traces from more than 31,000 users worldwide and different Android-based mobile device brands. By basing on this data-set, comparisons against a related approach published in [19] shows the superior performance of our model over 23 activity traces belonging to different mobile device users. This represents the major contribution of this paper w.r.t. our conference version presented at ITNG 2018 [20], where only one activity log was used to illustrate the proposed model. Other difference is the sampling range time used in the evaluation. While in the conference version the models were run for a single activity trace user over three months, in this version we included activity traces of 23 users for time ranges comprised between 13 and 25 months. Furthermore, contextual information of the applicability of the approach is presented in a new related works section.

The organization of the paper is as follows. Section 2 discusses relevant works related to mobile devices resource quantification, in the context of tasks scheduling mechanisms. These last constitute a vital component for exploiting the computing potential present in a dew cluster. Section 3 explains our model to predict energy availability in detail. Then, Section 4 evaluates the accuracy of the model in terms of the mean square root error metric, considering activity logs from several users form the device analyzer data-set and an alternative model from the literature [19]. Finally, Section 5 summarizes the implications of our findings and delineates future works and improvements.

2. Related Works

As stated in the introduction, mobile devices has been featured as a kind of dew computing device with great potential to cooperate in tasks execution within dew contexts. However, to perform a cooperation among several of such battery-driven devices, it is of utmost importance to develop indicators to quantify the resources availability of devices ahead in time.

In the context of scheduling mechanisms, quantification of resources availability has been tackled from different perspectives. One of these is the devices mobility perspective where can be found works focused on designing indicators to quantify the time a device is reachable by a central controller node from where tasks are distributed. For example, in [21] historical information of movements through different wireless coverage regions visited by a mobile device user, in working hours during a week, are summarized into a mobility score. The score is used by the proposed quantum-inspired binary gravitational search scheduler to rank devices according to their expected physical presence at a point of interest. Other forms of predicting physical presence is to use location change history in a Markov chain model [22]. The states of the Markov chain are the cells of the mobility area while transitions are possible cells which follow a given cell. Movements of a device from one cell to another are represented via a probability matrix whose values are updated every time a device moves. Predicting the next movement given a current location consists in retrieving the cell with the highest probability. In [23], a WLAN traces dataset from Dartmouth campus [24] is used to evaluate the performance of a resource quantification indicator called d-effectiveness, which includes mobile device parameters such as connection frequency and permanence time of devices with regard to different access points. Remaining battery data is not present in the validation since employed traces include wireless connection information only.

Complementary to mobile devices physical presence indicators is the quantification of resources associated to future energy availability. There is plenty room for improvement along this aspect [15], specially in correlation with device usage patterns which is the main goal of this paper. Several scheduling efforts limit to use battery percentage data, obtained via event-based battery APIs, to infer future energy availability, but ignoring information from device usage patterns that produce such battery changes. For instance, the *estimatedUptime* component of the simple energy-aware scheduler ranking formula [25] is a simple form of estimating the rate at which battery depletes. Using time and charge information of two consecutive battery events accessible via the Android Intents service, and assuming a lineal discharge rate between those samples, it is estimated the time at which battery will deplete. To adapt to the non-lineal discharge rate of a full battery discharge cycle the estimation should be adjusted (recomputed) with every new sample reported by the battery API [26]. In [27], the authors propose to construct a device energy model to forecast energy consumption for service selection operations. A service is represented by a demand vector where each component carries information of a resource type usage. The model for a device is obtained after running a set of benchmarks tests which exercise different resources usage combinations. By solving a lineal equation system derived from all tests, the energy consumption rate of each resource under consideration is obtained. Baseline energy consumption derived from OS processes are included in these rates.

All in all and after a careful analysis, we found approaches, in the context of tasks scheduling, that rely on synthesized information such as instant remaining battery events of mobile devices to determine future energy availability. The effectiveness of using device usage patterns derived from past owner activity for building a model that makes hour-wise predictions of battery information has been hardly explored, which motivates the proposal explained in the next Section.

3. Approach

Before diving into the details of our battery prediction model based on device usage patterns, let us first motivate, with two ordinary-life situations, the value of having a dew computing solution for hour-wise predictions of energy availability. Secondly, we illustrate how the model would be part of a dew computing architecture.

Drew and Jonathan own a company that offers their customers a service consisting in finding houses to buy and remodel. Every time a couple requests their services, Drew and Jonathan select a list of potential houses based on pre-requisites (budget, dimensions, proximity to desired places, and so on) and make an appointment with the couple to visit these houses. Drew and Jonathan employ an on-site software that is capable of interactively rendering remodeling options via augmented reality (AR), so a rich model is built as the camera of a smartphone films different parts and rooms of a house. The software is computationally expensive, and thus it might take advantage of nearby computing devices (smartphones) to interactively build the AR model. As houses being visited often have no electricity or internet connection, smartphones battery time must be both used wisely and collectively scavenged.

Under this scenario, it is desirable that if Drew and Jonathan will meet a couple at time t today, the available battery lifetime of both their smartphones and that of the target couple are predicted in advance based on how each user employs their device. This way, not only the available computing power to use the AR software at time t is known beforehand, but also the various smartphones are wisely assigned parallel tasks upon rendering remodeling options, so the remaining battery after all planned houses are visited is affected evenly among the participating smartphones. For obvious reasons, the dew scheduler however might decide to put more burden—in terms of assigned task to execute—on Drew and Jonathan's mobile phones rather than on those of their clients.

Let us consider another relevant situation. A tourism company offers customers a trekking service on beautiful but isolated geographic places across a country. Expeditions to a particular place depart at predefined days and hours (e.g., t) from a fixed meeting point, and the time required to reach a place from the meeting point is known beforehand (e.g., r hours). Again, internet connectivity in such places is absent. Moreover, the company provides customers with a novel AR software that allows users to augment the scene with information to enhance the trekking experience; as the user points its smartphone to local vegetation, the software uses image recognition algorithms and images with information and interactive animations regarding e.g., in which animals' food chain a specific vegetation is. Similar to the previous situation, the dew scheduler associated to the software might predict battery lifetime in participating smartphones at $t + r$ on a specific day, and hence distribute parallel tasks based on this information. In this case, the scheduler might also want to put the computing burden on some smartphones, while saving battery on few others due to emergency calls that might need to be issued.

Figure 2 illustrates asynchronous interactions in time between components of a dew-cloud architecture instantiated for our approach. From time t to $t + \alpha'$, a synchronization/training phase takes place, while $t + \alpha''$ moment corresponds to dew devices cooperating, i.e., a dew computing phase. Concretely, at time t, a dew computing device synchronizes with the cloud and the last registered usage pattern data chunk is derived from past owner activities. Chunks data are relevant features that we will detail next in this section. Due to limited storage capacity, the dew device might be configured to save up to a certain number of past chunks, then, a synchronization operation allow the device to erase old chunks. Cloud servers, however, have a large capacity to store many chunks, i.e., long periods of owner activity which would be used to (re)train an individual battery prediction model for the dew device owner. At time $t + \alpha$, while the dew computing device continues registering mobile device owner activity locally, the cloud trains a battery prediction model. At $t + \alpha'$, upon a new dew device-cloud synchronization, the dew device gets a trained battery model from the cloud.

Now, let us suppose the above component's interaction can be replicated by any number of dew devices simultaneously. Besides, the training result of each dew device prediction model could be different, in the sense that one could generalize better the battery behavior than others. Finally, at $t + \alpha''$, is represented dew computing scenarios when a cluster of dew devices are required to cooperate, for instance, for executing tasks of a compute intensive application like the augmented reality scenarios described above. Cooperation is performed with future battery information obtained e.g., via a micro service running in the dew device, which invokes an instance of our battery prediction model.

Having said so, in subsequent sections, we focus on explaining the underpinnings of the proposed battery prediction model, which is the focus of this paper. In summary, our approach includes an ensemble of several Machine Learning algorithms trained using features resulting from a feature selection analysis performed over a subset of user activity events. Activity events derived from traces of real mobile device users logs. Features belong to different categories including energy (e.g., when the user plugs/unplugs the mobile phone from a wall socket or USB), network (e.g., Wifi scanning/connection/disconnection events) and application (e.g., screen on/off, launch/close an application).

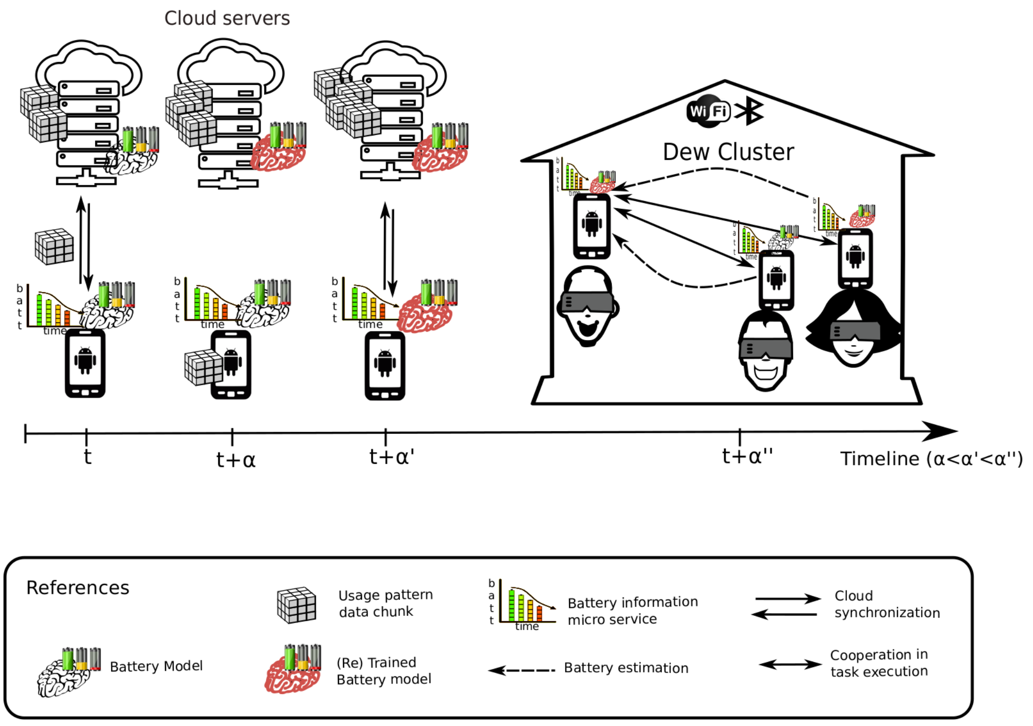

Figure 2. Battery prediction model and its relation with dew computing architecture: schematic overview.

3.1. Preliminary Data Analysis and Feature Selection

In machine learning, feature selection is the process of defining a subset of relevant features (variables, a.k.a predictors) for use in a model construction. Feature selection pursues two goals, namely reducing the number of features to avoid overfitting/improve the generalization of models built, and gaining better understanding of the features relationship to the response variables [28]. To come up with a feature selection approach, general enough in the problem domain at hand and built based on real user data at the same time, we utilize the Device Analyzer data-set [18]. This is provided by the University of Cambridge, and represents to date the largest collection of activity traces from real Android-powered mobile device users. At the time of writing this paper, this data-set has 31,455 contributors worldwide.

For the sake of illustrating the feature selection approach, values shown below correspond to a single user from the device analyzer data-set, who has registered activity traces for over 6 weeks. In order to ease the analysis of the user activity data, which basically consists in per-user files with registers composed by an id field, milliseconds, a time-stamp, a data field typically represented by a data category, subcategories and values associated, we firstly defined a structured format for it. To this end, the data-set was split into states, where each state has the minute-wise value for each mobile sensor in a device. A mobile device state changes as a result of an event in time that changes the value of any sensor. For example, a change in the battery level can be defined as an event, which triggers a new state of the mobile device in which the battery level is modified and all other features remain the same. Initially, from the formatted data, the following combination of features were considered:

- Day of week: with values from 0 (Sunday) to 6 (Saturday) indicates the week day where the event was registered. Type: integer.
- Minute: with values from 1 to 1440 that represent the minute of the day in which the event occurred. Type: integer.
- External supply: Takes a 1/0 value and indicates whether the device is plugged to an external energy supply, e.g., AC adapter or USB connection, or not respectively. Type: Boolean.
- Brightness level: with values from 0 to 100 indicates the screen brightness percentage intensity. Type: integer.

- Screen on/off: takes a 1/0 value and indicates whether the device screen is active or inactive respectively. Type: Boolean.
- Connected: takes a 1/0 value indicating whether the device is connected to a 3G/4G network or not respectively. Type: Boolean.
- Connected to Wifi: takes a 1/0 value indicating whether the device is connected to a Wifi network or not respectively. Type: Boolean.
- Temperature: indicates battery temperature. Type: Integer.
- Voltage: indicates battery voltage. Type: integer.
- Battery level: with values from 0 to 100, indicates remaining battery level. Type: integer.

The data-set itself contains several other features, such as those related to location and application usage. Figure 3 shows a raw extract of few registers from an activity trace.

```
997;49603804;2013-02-03T10:19:49.368+0400;app|end;
998;49621946;2013-02-03T10:20:07.510+0400;power|battery|scale;100
999;49621947;2013-02-03T10:20:07.511+0400;power|battery|level;14
1000;49621947;2013-02-03T10:20:07.511+0400;power|battery|temperature;310
1001;49621947;2013-02-03T10:20:07.511+0400;power|battery|voltage;3620
1002;49640676;2013-02-03T10:20:26.240+0400;conn|WIFI|detailedstate;CONNECTED
1003;49640677;2013-02-03T10:20:26.241+0400;wifi|connected|810bb9a2d8c43074d34fef2a7a8d2d3fad9e3558|ssid;ed6a32d36431578409119ed41f
1004;49640677;2013-02-03T10:20:26.241+0400;wifi|connected|810bb9a2d8c43074d34fef2a7a8d2d3fad9e3558|rssi;-53
1005;49640677;2013-02-03T10:20:26.241+0400;wifi|connected|810bb9a2d8c43074d34fef2a7a8d2d3fad9e3558|linkspeed;54
1006;49640683;2013-02-03T10:20:26.247+0400;net|wlan0|rx;798022341
```

Figure 3. Raw extract from a device analyzer activity trace.

Nevertheless, these features were not considered as relevant to the battery level modeling as the ones previously listed. First of all, it is true that by knowing the location of users in each state, it might be possible to infer if they are at home. If that is the case, the probability of charging the mobile phone increases. However, that information can also be obtained from the charging pattern using the energy supply feature. Secondly, application usage is highly seasonal, because some applications tend to be used in a certain period of time (e.g., flight-support or tourism applications). In addition, users might also change their applications frequently due to e.g., updates or replacements. Therefore, it is very difficult to generalize a model based on this feature, and it might incur in extra noise.

Table 1 shows an extract from the formatted data-set for one particular mobile device user, i.e., the one used in [20] to illustrate the model. Furthermore, Figure 4 depicts the battery level variation along time for this user (first 5 days of the sample). It is possible to see that the resulting curve is not exactly periodic since it takes different shapes, and it does not have a pre-set behavior. However, it is worth noting that there is a visual resemblance to a sinusoidal curve, because it continuously goes up and down. In fact, Figure 5 (left) depicts a greater resemblance when the battery level is averaged per day. Considering this observation, which applied to many other users in the data-set as well upon producing preliminary visual representations of battery usage, a new feature representing the sinus movement was added to the feature list:

$$Battery\,sinus = amplitude \times sin(2 \times pi \times \frac{minute}{minsPerDay}) + batteryLevelMean \tag{1}$$

where amplitude describes how much the curve goes up and down, *minute* is the minute of the day, *minsPerDay* is the total number of minutes in a day (i.e., 1440), and *batteryLevelMean* is the average battery level per day. The *batteryLevelMean* is the mean per day battery level in the data-set. In addition, the maximum amplitude is 50, since a bigger value makes the curve going above the maximum battery level or below the minimum battery level. Finally, it is worth pointing out that the only variable used to calculate the sinus is minute, which is easily accessible in every mobile device. Figure 5 (right) shows the resulting curve compared to the real one (in the left) we previously obtained using real battery level samples.

Table 1. Extract from the formatted data-set for an individual user. Cells in Boolean-typed columns contain either 0 (false) or 1 (true).

Day of Week	Minute	External Supply	Bright Level	Screen on/off	Connected	Connected to Wifi	Temperature	Voltage (mV)	Battery Level
3	652	0	149	1	0	1	275	3686	31
3	653	0	149	1	0	1	286	3740	30
3	654	0	149	1	0	1	286	3740	30
3	655	0	149	1	0	1	286	3740	30

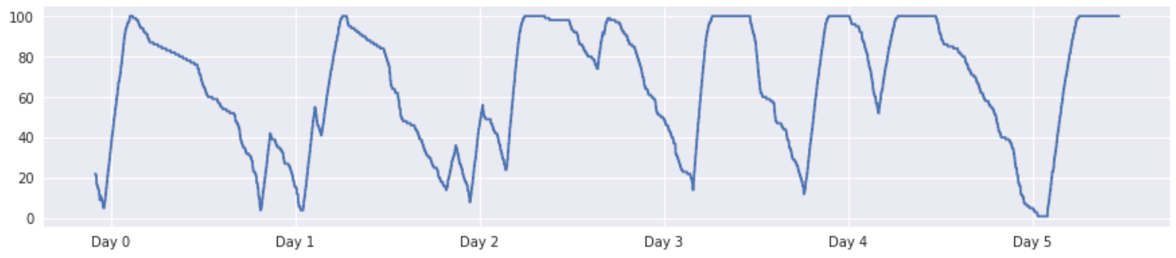

Figure 4. Battery level overview of the sample user in the first five days.

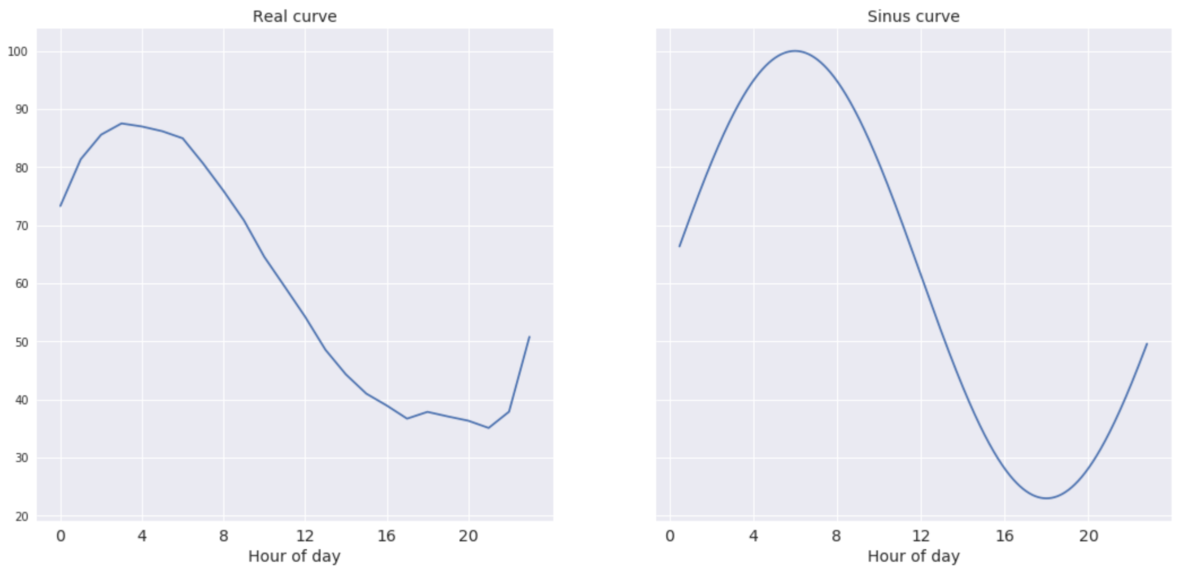

Figure 5. Per-hour average battery level of the sample user: real (left) and sinus curve (right).

Likewise, as Figure 4 shows, during night hours battery level tends to go up, which means that the device remains plugged to an energy supply. Such battery level pattern, i.e., the zenith at night when the device is usually connected, and the decrement during daytime caused by the device usage can be modeled with a cosine function. Concretely, based on this reasoning, it was included the extra feature which appears in Equation (2):

$$Battery\,cosine = amplitude \times cos(2 \times pi \times \frac{minute}{minsPerDay}) + batteryExternalMean \tag{2}$$

In this case the amplitude has a maximum value of 0.5, because the energy supply can only take values 0 or 1. Then, *batteryExternalMean* is the average of all the external supply feature values in the user trace data. Figure 6 (right) shows the curve obtained from this calculation, and Figure 6 (left) shows the real averaged curve.

In addition, to capture the battery variation (derivative) from event to event, we added a previous battery level feature to the feature set, given by the battery level in the previous sampling time-step. Table 2 outlines descriptive information of all user activity traces studied in this work whose formatted data was preprocessed through the procedure described above. Particularly, third, fourth and fifth columns show that heterogeneity is present in aspects such as device brand and model, sampling time range, and amount of registers resulting from the preprocessing of the formatted data.

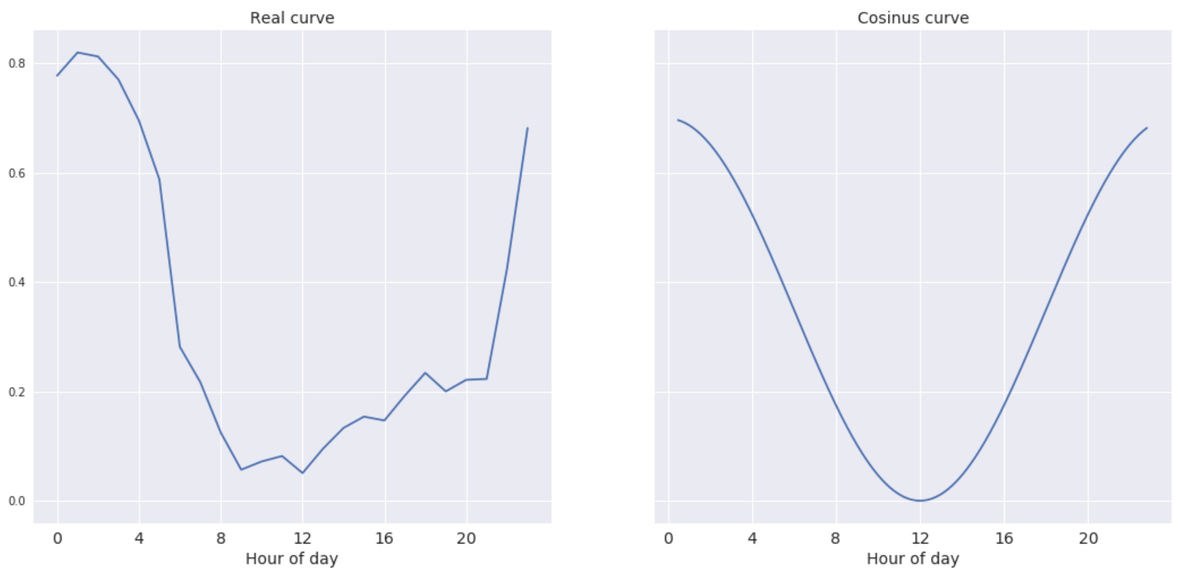

Figure 6. Per-hour average energy supply feature value on sample user data-set: real (left) and cosine curve (right).

Table 2. Data-set descriptive information.

Device-Analyzer id	**Paper User id**	**Device**	**Sampling Time Range**	**Preprocessed Samples**
2ed95c7b731796c93e6b3d64838999c544cd9a3c	User #1	LG E970	2013-07-07/2014-10-26	562,309
610556db933b1efc49102b24d25a38ea4046fa83	User #2	Asus Nexus 7	2013-03-04/2014-06-06	651,011
637085b6a0f994e3553c04e1fb1ba34adc79b45f	User #3	LG Nexus 7	2013-06-25/2015-04-21	1,214,104
776b987e3603dc29d0e69ab02589495d16ef4ab0	User #4	Samsung GT-I9300	2013-04-17/2015-04-17	654,729
eb479ee4954f6cfdac96e7f96da8082e9ed14448	User #5	Samsung Galaxy Nexus	2012-08-08/2013-09-03	570,516
20c3e02df67c67e03c25d9e652e6824223566e97	User #6	Sony C6603	2013-09-04/2014-12-17	655,531
30d303a9bd7fe9b25843221664ed0a06327513cc	User #7	Asus Nexus 7	2013-04-09/2015-03-30	920,499
35b1d2e2fba71ffbd464e9332694449ebb4c6abc	User #8	Sony C6802	2013-10-10/2015-04-22	626,555
3894eeb7d815d317b03832230d789ea5f0976431	User #9	Samsung GT-N7105	2013-05-22/2015-04-16	491,890
3f76e9998242f8d1deccf8768317de51f8016dfc	User #10	Samsung GT-I9300	2012-10-04/2014-03-04	1,166,769
43a0307073bbffb25fc8a61b9208984643542fd5	User #11	Sony C6603	2013-09-13/2014-12-03	477,212
47489d0baa333944707e7e73eddf7217e3b6ad6f	User #12	Samsung Nexus 10	2013-03-02/2015-04-21	1,084,742
4e35914592394bd77618bf5fbb082c5633e505b3	User #13	Samsung SGH-I337	2013-04-30/2015-04-20	1,763,679
97c303108eafe034b9abd3da8e99de69c75b007c	User #14	Samsung Nexus 10	2013-09-15/2014-12-21	644,837
a70bf30cf27a42de6fea303ce5741175d1576db2	User #15	Samsung GT-I9300	2013-04-16/2014-06-15	1,187,940
ad6805eaa1c029bd984787cfae32ba08a31eb760	User #16	Samsung Galaxy Nexus	2012-06-01/2014-01-11	837,551
aebdbfd3f63250921dc09ad26d4bf53560744f71	User #17	LG Nexus 5	2013-11-13/2015-04-22	767,949
b73b55214a94fa6820e66004953b3d78cf4e55cc	User #18	Asus Nexus 7	2013-05-18/2015-03-18	238,914
d7c99d131faa1b6127d66298b6f2dc8b399799f5	User #19	Sony LT29i	2013-02-15/2015-01-24	2,886,243
e0f8b15eac51a415ce7c8ef3b7bc44c7f742c81b	User #20	Motorola XT1032	2014-03-19/2015-04-21	491,097
f61dac311bac0ecd8e0ce49f20d8982f07c1ccff	User #21	Asus Nexus 7	2013-09-15/2015-02-06	694,249
f76e8746e60aeec21805ddde542219068dd03999	User #22	Samsung GT-I9300	2013-11-15/2015-04-22	607,084
ff3925bbbe0bb0b08af35f3b997f452d82ab71ed	User #23	Sony C6603	2013-11-20/2015-04-02	628,723

3.1.1. Correlation Analysis

Taking into account the features associated to any mobile user activity with his/her device, we performed a correlation analysis between these using the Pearson correlation coefficient. The analysis was made on the basis of the sample user already selected to illustrate the approach.

The analysis indicated that there is a highly strong positive correlation between battery level and voltage. This means that voltage is not discriminant for our model and it will overfit any Machine Learning model, hence it should not discarded for the steps of the approach which follow. Likewise, the temperature should also be discarded (highly negative correlation with the battery level).

Secondly, the analysis showed a highly strong negative correlation between connected and connected to Wifi. This means that when the device is connected to Wifi, 3G/4G is not used, and the same applies in the opposite direction. This might be because Android turns off 3G/4G connection when there is a Wifi available in order to avoid wasting mobile data quota, which costs money.

Another interesting outcome to point out is between the sinus feature and the battery level, which is interesting to show that the model can actually rely on this feature to predict the battery level for future states. Although slightly weaker, there is a noticeable positive correlation between the external supply and the cosine feature.

Finally, we found that the screen tends to be off when the external supply is connected. Probably, the mobile device was left charging while hardly interacting with it in our sample user. Besides, there is a negative correlation between the hour and the battery level, meaning that the mobile device has less remaining energy as time passes by. This is consistent with the curve shown in Figure 4.

3.1.2. Feature Selection

After stages of preprocessing the data-set, we refined the feature selection criterion to derive a descriptive feature set with as few features as possible. At this stage, features that are not influential in the battery level are ruled out. As the literature suggest, feature selection can be supported via many strategies [28]. To obtain more accurate results, we decided to use a combination of three different techniques.

Firstly, two univariate feature selection [28,29] techniques were considered: F-regression and mutual information. F-regression is useful to determine/estimate, given two random variables, its degree of linear dependency. To exploit F-regression, we determined linear regression between each variable against the battery level. Secondly, mutual information is a non-parametric technique that uses entropy to assess information content in data, considering that higher entropy levels means lower information content and vice-versa. Thirdly, we used the Lasso score, which is a regression model with $L_1 = |w_i|$, regularization term, where w_i is the coefficient used for x_i. Unlike the well-known ridge regression regularization term [30], Lasso shrinks those "irrelevant" coefficients to zero rather than shrinking them to negligible values. The Lasso score prescribes a meta-parameter *alpha*, which was set to 0.0001 in our experiments. The larger the *alpha*, the more the features were discarded. With *alpha* equals to zero, Lasso behaves as a regular linear regression.

Table 3 depicts the values from F-regression, mutual information and Lasso for the sample user. The first column shows the F-regression score between each feature and the battery level, which is the target feature. The second column depicts the mutual information score derived from the entropy of the data-set. The third column shows Lasso scores, or the individual coefficients for each feature obtained after training the regression model with the regularization term.

Table 3. Feature selection based on F-regression, mutual information and Lasso: scores for the sample user.

	F-Regression	**Mutual Information**	**Lasso**
Previous Battery Level	3.86×10^9	3.995	0.9966
Minute	2.06×10^5	0.599	0.0088
Day of Week	5.25×10^2	0.067	0.0002
External Supply	1.28×10^5	0.193	0.467
Connected	1.21×10^3	0.028	0
Connected to Wifi	2.85×10^3	0.024	0
Bright Level	2.87×10^4	0.002	0.0002
Screen On/Off	3.22×10^4	0.059	0.007
Sinus	2.61×10^5	0.308	0.086
Cosine	6.39×10^3	0.244	0

3.2. Model Construction

In this section, we explain the proposed model to predict mobile phones battery level in future states. In particular, it is a result of combining classification models and a regression model to predict a new state based on a previous one. First of all, the selected model to estimate the battery level is a multiple linear regression model built using the resulting features from previous section. Multiple linear regression models are the most suitable machine learning algorithms for predicting a continuous variable, such as mobile phones battery level, depending on a set of different features. Moreover, the model was trained using 10 percent of a user profile gathered data, and the rest 90 percent was used for testing purposes.

In addition to the regression model, two complementary classification models were built to estimate the variability of external supply and screen on/off features. These models predict the state of both features in future states. The external supply classifier is a random forest classifier with 10 estimators and its maximum depth was three. It was trained using the cosine as well as the minute of the day that is being predicted and the previous state. The screen on/off estimator is a decision tree classifier built using the minute of day and the previous screen state.

The proposed mechanism to combine all the created models is an iterative method that estimates the following state based on the previous estimated states. The pseudocode of the proposed method is shown in Algorithm 1.

Algorithm 1 Pseudo-code of the remaining battery prediction method

```
procedure PredictFutureRemainingBattery($desired\_state$)
    while $current\_state < desired\_state$ do
        $current\_state[external\_supply] \leftarrow external\_supply\_classifier$.ESTIMATE($current\_state$)
        $current\_state[screen] \leftarrow screen\_classifier$.ESTIMATE($current\_state$)
        $current\_state \leftarrow regression\_model$.ESTIMATE_FOLLOWING_STATE($current\_state$)
    end while
end procedure
```

In this context, a desired state is a particular minute in the near future where it is desired to know the battery level. This becomes pivotal in the context of dew scheduling, where the scheduler should consider, among several variables, how much available energy a mobile device will have to execute specific tasks. By being able to know the device's future battery level, the scheduler would be aware of future resource exhaustion, and then could avoid assigning a task to a device that, for example, will run out of battery while executing [15].

4. Evaluation

In this section, we describe the experiments performed to evaluate the model proposed in the previous section. Specifically, instances of the estimation model is trained using the activity traces of 23 users, and then, the resulting model for each user is compared to Kang et al. method, a similar approach from the literature [19] operating on the same training data. To assess the prediction results and compare the approaches we used the Mean Squared Error metric, which is defined as:

$$MSE = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2. \tag{3}$$

Another alternative metric we could had employed is MAE (mean absolute error), but in this case we want to penalize more those estimations that are further away from the real mean value, such as wrongly estimating that the battery level would be much below or over the real battery level. Note that in the ITNG 2018 version of this paper [20], we evaluated our approach using traces from a single user only.

With respect to the Kang et al. method proposed in [19], is based on predicting the battery level via an approach that defines all the possible combinations of sensor states, such as energy supply connected and Wifi connected, energy supply connected and Wifi disconnected, and so on. After that, the method figures out the average time a user spends on each state, and the average battery consumption per state. Finally, the method computes the battery level by feeding that information to the following formula:

$$BatteryLevel = T \times \sum_{i=1}^{N} p_i \times B_i \tag{4}$$

where T is the number of minutes ahead for which the battery level is to be predicted, p_i is the average number of minutes spent on state i and B_i is the average battery consumption per minute.

Regarding the test setup, both models are trained using the first 10% samples of a user's trace data. Then, battery predictions were made for each of the following days in the data-set. Particularly, we picked an hour of the day and run each model to estimate the battery level for the next hours. After that, the mean squared error (MSE) was calculated for each curve and averaged to get the MSE per day for each model. The selected hour of the day was 12 p.m. because it is the time of the day on which the mobile users have more activity, and it is when dew schedulers might take more advantage of the model because of the consequent abundance of active mobile devices. Besides, making a prediction of five hours ahead is a good baseline for comparing both approaches, since [19]'s model does not aim to return good estimations in the long run. This can be quickly visualized in Figure 7, where estimations using both models for our sample user are depicted.

The final result of this process can be found in Table 4 for 23 users from the device analyzer data-set. The columns show the average MSE per week day at 5 p.m. for both methods. In bold is highlighted the method that achieved the minimum MSE in the experimental scenario. The latter is defined as a prediction made on a week day at 12 p.m. five hours ahead. Refer to Table A1 in the Appendix A for information of the amount of samples involved in the computation of each MSE value. It is clear that our proposed method achieved the lowest MSE in 136 out of 161 prediction scenarios, considering an instance in the latter the prediction was made for a weekday-hour combination. This outcome is due to the fact that the Kang et al. method is based only in the average consumption, and it does not take into account time-related factors, such as when the battery is going to be charged. Our proposed method, instead, not only takes into account previous states by learning its behaviour with a regression model, but also predicts whether the mobile phone is going to be connected to the energy supply or not in the future. That feature helps the model to figure out if the series is going up or down at a specific time. Besides, by predicting whether the screen is going to be on, the model can adapt the fastness at which the battery level decreases or increases (depending on the energy supply).

Table 4. Per-day average mean squared error (MSE) of Kang et al. [19] and our approach.

	Sunday at 5 p.m.		**Monday at 5 p.m.**		**Tuesday at 5 p.m.**		**Wednesday at 5 p.m.**		**Thursday at 5 p.m.**		**Friday at 5 p.m.**		**Saturday at 5 p.m.**	
User id	Kang et al.	Ours	Kang et al.	Ours	Kang et al.	Ours	Kang et al.	Ours	Kang et al.	Ours	Kang et al.	Ours	Kang et al.	Ours
User #1	**9.1**	15.2	**11.4**	18.4	**10.2**	16.5	**11.6**	18.7	**9.7**	17.2	**13.0**	16.1	**14.9**	19.3
User #2	23.0	**20.7**	**19.1**	21.2	23.0	**22.2**	20.7	**18.9**	**17.3**	18.7	24.7	**24.4**	**18.5**	22.6
User #3	**14.2**	19.4	**14.1**	20.4	**17.4**	21.1	**18.6**	20.4	**12.1**	16.6	**17.3**	24.4	**21.0**	22.5
User #4	**21.2**	26.3	**18.6**	23.2	**18.2**	19.2	**16.6**	20.9	**17.0**	20.1	25.0	**20.2**	**18.2**	20.3
User #5	26.7	**24.0**	29.4	**26.3**	**23.7**	26.7	33.2	**32.4**	**30.2**	33.9	30.9	**28.3**	37.0	**31.1**
User #6	35.2	**14.1**	36.7	**12.6**	32.5	**12.2**	35.0	**13.2**	34.0	**13.0**	25.9	**15.0**	25.6	**16.0**
User #7	55.9	**13.4**	54.2	**14.1**	55.3	**16.6**	50.2	**14.1**	54.2	**15.2**	54.3	**16.0**	53.1	**12.8**
User #8	69.4	**19.1**	67.9	**19.8**	71.8	**18.2**	72.1	**17.7**	73.3	**19.8**	71.4	**18.2**	72.2	**18.0**
User #9	55.7	**12.9**	59.8	**14.1**	60.6	**13.8**	57.9	**10.7**	60.9	**14.6**	54.5	**12.4**	59.3	**17.9**
User #10	26.3	**16.9**	26.9	**16.0**	30.8	**15.0**	32.4	**16.4**	30.9	**14.0**	24.7	**16.3**	32.3	**16.7**
User #11	54.9	**29.1**	55.2	**34.3**	54.9	**30.8**	53.5	**29.8**	49.4	**31.7**	54.2	**27.9**	49.3	**30.3**
User #12	37.4	**10.9**	38.5	**11.1**	38.6	**13.2**	36.8	**11.4**	38.0	**10.5**	42.4	**9.3**	42.8	**13.2**
User #13	25.7	**14.9**	21.9	**16.7**	20.4	**16.8**	22.3	**16.6**	18.6	**18.3**	25.7	**17.3**	22.0	**17.5**
User #14	31.6	**15.7**	36.5	**13.6**	40.8	**14.6**	37.3	**15.9**	36.9	**14.9**	38.5	**16.4**	40.3	**21.3**
User #15	36.9	**19.5**	40.1	**21.5**	37.1	**20.2**	37.4	**20.2**	35.9	**17.3**	45.6	**22.2**	47.6	**29.3**
User #16	45.9	**19.7**	42.9	**25.1**	45.5	**20.2**	42.1	**26.6**	46.2	**17.6**	46.1	**13.2**	40.5	**20.0**
User #17	32.7	**21.3**	38.1	**17.7**	35.6	**19.1**	37.6	**17.5**	33.2	**21.0**	33.6	**20.2**	33.1	**22.7**
User #18	30.1	**10.6**	30.0	**9.3**	28.1	**9.2**	31.2	**10.4**	29.6	**14.7**	28.9	**8.3**	28.2	**11.8**
User #19	29.4	**18.0**	29.7	**15.5**	29.1	**19.7**	30.3	**20.3**	29.7	**19.9**	26.8	**22.0**	30.7	**23.3**
User #20	28.9	**12.7**	26.2	**14.3**	26.4	**14.5**	26.9	**16.4**	30.6	**14.6**	31.0	**14.1**	32.5	**13.6**
User #21	37.0	**10.8**	43.5	**9.2**	48.9	**11.2**	40.5	**12.3**	42.1	**12.0**	42.2	**14.3**	47.1	**15.6**
User #22	32.2	**13.8**	32.6	**11.5**	31.8	**14.7**	33.8	**13.3**	29.2	**14.7**	30.6	**14.0**	28.0	**14.9**
User #23	42.5	**25.4**	42.1	**27.1**	43.4	**24.1**	47.4	**22.8**	43.6	**26.6**	48.0	**25.0**	41.2	**28.2**

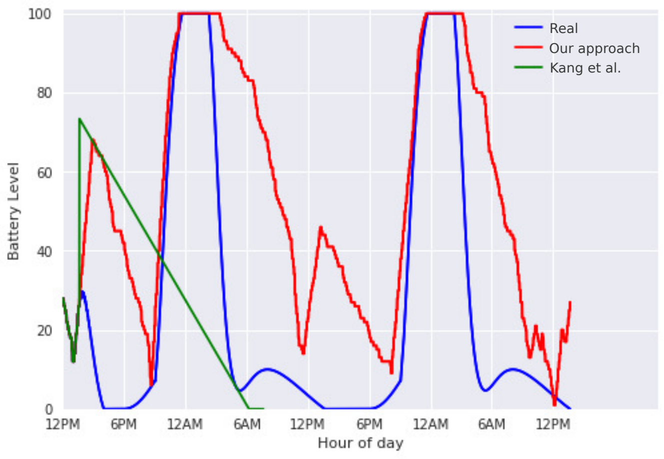

Figure 7. Predicted battery level for a period of 48 h using the proposed approach and [19].

The complementary models make our approach more precise for longer periods of times. Figure 7 shows the predicted battery level for a period of 36 h using the proposed approach, i.e., from a given starting point it iteratively predicted, state by state, the final battery level after 48 h. It is clear that there is a significant visual resemblance between the real battery level and the estimated one. Moreover, the energy supply model had a very high influence in predicting the hour when the battery level was going to increase. For that particular case, Kang et al.'s MSE was 57.67, while our approach obtained 27.86.

5. Conclusions and Future Directions

In this paper we started by contextualizing within dew computing scenarios, a novel model to predict battery availability in mobile devices. The prediction model exploits past device owners activity and relevant device state variables via a two-phase approach, which includes feature extraction/selection techniques on one hand and regression models and machine learning classifiers on the other hand.

Preliminary experiments performed using mobile phone usage data from 23 different users from the device analyzer data-set and comparisons against the estimation model published in [19] yielded encouraging results. Our model was able to reduce the MSE metric in all experimental scenarios for 18 out of the 23 users, and hence estimations are more accurate than the competitor approach. This is in line with our utmost objective, which outfit scheduling mechanisms with long term future energy availability information in order to consider mobile devices as first-class resource providers in dew computing scenarios [17]. It is worth noting that we have a preliminary implementation of our model, which was done in Python using open-source libraries commonly used in data science and machine learning applications applications (Scikit-Learn, Pandas, Numpy and MatplotLib). The source code is available from a GitHub repository (http://github.com/matlongo/battery-level-predictor).

Even when the device analyzer data-set contains many users with activity data spanning several weeks and even months, as indicated earlier, there are many activity events missed. This is due mainly to the device analyzer application being unable to register events for certain periods of time, and the user shutting down the mobile device during several hours and even days. We took a simplistic approach to data preprocessing in order to circumvent such cases (e.g., we filled missing battery samples assuming the battery level remained constant during such gaps). Then, in the future, more elaborated models to fill in the potential missing activity data must be developed. In addition, other machine learning models can be explored to capture undiscovered relationships between features, and in this way improve predictions accuracy. Specifically, we will analyze a recurrent neural network (NN) known as long short term memory (LSTM) [31]. Unlike traditional feed-forward NN, recurrent NN have memory and hence predictions depend on previous states.

Moreover, LSTMs are a special kind of recurrent NN which is very effective for predicting state sequences and pattern while considering the time dimension [32], hence all previous states can be taken into account. Finally, we could also compare our results against battery estimation models from the industry, such as those behind highly-popular battery manager applications in the Google Play Store such as Battery Doctor and Battery Pro Saver.

Other possible extension is to outfit the proposed models with information derived from a time series analysis. The Device Analyzer Android app collects samples in a device at constant time intervals, which derive in a collection of data points (user profile) ordered chronologically. Based on this and on that battery level is time dependent, a seasonal dependency analysis could be applied to identify relations between i-th and $(i–k)$-th data points. Intuitively, it can be inferred that if a time series has a particular behaviour over time, there is a very high probability that it will follow the same in the future. For the purpose of this work, k, also called lag, is the number of previous states that are needed to come up with a good battery level estimation. Since intuitively estimations have a strong dependence on previous state's values, it is necessary to know how many previous states could help to improve the prediction of future battery level. The application of an autocorrelation function (ACF) and partial autocorrelation function (PACF) are commonly used to examine seasonal patterns in time series.

Finally, we will integrate our battery predictor with the dew schedulers we have already developed [17]. These schedulers operate via energy-aware indicators that, given some ready-to-execute parallel tasks in a dew application, indicate which are the most suitable mobile devices to execute the tasks. At present, battery estimation schemes underpinning these indicators are fairly rudimentary, so a long-term battery estimator will give added value to our schedulers.

Author Contributions: Conceptualization, M.H. and C.M., M.L.; methodology, C.M.; software, M.L.; validation, M.L., M.H.; formal analysis, M.L., M.H.; investigation, M.H., C.M., M.L.; resources, A.Z.; data curation, M.L.; writing–original draft preparation, C.M., M.H.; writing–review and editing, M.H., C.M.; funding acquisition, A.Z. and C.M.

Funding: This research was funded by ANPCyT and CONICET through grants PICT-2013-0464 and PIP no. 11220170100490CO respectively.

Acknowledgments: The first author acknowledges his MSc. scholarship in Data Science (USA) granted by Fundación Sadosky. The second author acknowledges his Postdoctoral scholarship granted by CONICET.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1 outlines the number of predictions that could be made for each experimental scenario. Variations for a user stem from the fact that user activity traces present gaps of missing information for which no data filling strategy could be applied. Missing data for the selected day of week/hour combination does not allow us to validate predictions so that these had to be discarded.

Table A1. Per-day predictions made in each experimental scenario.

User/Day	Sunday at 5 p.m.	Monday at 5 p.m.	Tuesday at 5 p.m.	Wednesday at 5 p.m.	Thursday at 5 p.m.	Friday at 5 p.m.	Saturday at 5 p.m.
User #1	47	44	50	48	51	49	45
User #2	21	25	21	24	24	25	23
User #3	57	61	63	65	55	81	62
User #4	33	43	43	50	47	46	44
User #5	44	39	35	46	42	44	48
User #6	49	50	49	57	54	47	34
User #7	61	75	73	62	70	64	63
User #8	44	45	46	51	47	41	38
User #9	40	36	42	42	36	36	34
User #10	48	44	53	46	40	54	61
User #11	33	41	49	35	37	45	46
User #12	60	61	63	61	59	60	69
User #13	51	62	57	56	65	56	44
User #14	43	42	42	44	47	46	55
User #15	55	53	55	54	59	66	41
User #16	73	73	64	65	74	67	58
User #17	54	57	51	57	46	61	55
User #18	17	21	22	20	20	20	25
User #19	92	102	92	90	97	79	89
User #20	32	33	35	38	37	40	42
User #21	51	53	50	48	45	48	48
User #22	46	42	51	41	45	48	45
User #23	52	52	50	49	49	60	56

References

1. Skala, K.; Davidovic, D.; Afgan, E.; Sovic, I.; Sojat, Z. Scalable distributed computing hierarchy: Cloud, fog and dew computing. *Open J. Cloud Comput.* **2015**, *2*, 16–24.
2. Nunna, S.; Kousaridas, A.; Ibrahim, M.; Dillinger, M.; Thuemmler, C.; Feussner, H.; Schneider, A. Enabling real-time context-aware collaboration through 5G and mobile edge computing. In Proceedings of the 12th International Conference on Information Technology-New Generations, Las Vegas, NV, USA, 13–15 April 2015; pp. 601–605.
3. Vaquero, L.M.; Rodero-Merino, L. Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 27–32. [CrossRef]
4. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; pp. 13–16.
5. Ahmed, E.; Ahmed, A.; Yaqoob, I.; Shuja, J.; Gani, A.; Imran, M.; Shoaib, M. Bringing computation closer toward the user network: Is edge computing the solution? *IEEE Commun. Mag.* **2017**, *55*, 138–144. [CrossRef]
6. Ray, P.P. Minimizing dependency on internetwork: Is dew computing a solution? *Trans. Emerg. Telecommun. Technol.* **2019**, *30*, e3496. [CrossRef]
7. Ray, P.P. An Introduction to Dew Computing: Definition, Concept and Implications. *IEEE Access* **2018**, *6*, 723–737. [CrossRef]
8. Wang, Y. Cloud-dew architecture. *Int. J. Cloud Comput.* **2015**, *4*, 199–210. [CrossRef]
9. Fisher, D.E.; Yang, S. Doing more with the dew: A new approach to cloud-dew architecture. *Open J. Cloud Comput.* **2016**, *3*, 8–19.
10. Gusev, M. A dew computing solution for IoT streaming devices. In Proceedings of the 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 22–26 May 2017; pp. 387–392.

11. Martins, V.; Rufino, J.; Silva, L.; Almeida, J.; Miguel Fernandes Silva, B.; Ferreira, J.; Fonseca, J. Towards Personal Virtual Traffic Lights. *Information* **2019**, *10*, 32. [CrossRef]
12. Rawassizadeh, R.; Pierson, T.J.; Peterson, R.; Kotz, D. NoCloud: Exploring network disconnection through on-device data analysis. *IEEE Pervasive Comput.* **2018**, *17*, 64–74. [CrossRef]
13. Curiel, M.; Calle, D.F.; Santamaría, A.S.; Suarez, D.F.; Flórez, L. Parallel Processing of Images in Mobile Devices using BOINC. *Open Eng.* **2018**, *8*, 87–101. [CrossRef]
14. Harwood, A.R.; Revell, A.J. Parallelisation of an interactive lattice-Boltzmann method on an Android-powered mobile device. *Adv. Eng. Softw.* **2017**, *104*, 38–50. [CrossRef]
15. Hirsch, M.; Mateos, C.; Zunino, A. Augmenting computing capabilities at the edge by jointly exploiting mobile devices: A survey. *Future Gener. Comput. Syst.* **2018**, *88*, 644–662. [CrossRef]
16. Tapparello, C.; Karaoglu, C.F.B.; Ba, H.; Hijazi, S.; Shi, J.; Aquino, A.; Heinzelman, W. Volunteer Computing on Mobile Devices: State of the Art and Future. In *Enabling Real-Time Mobile Cloud Computing through Emerging Technologies*; IGI Global: Hershey, PA, USA, 2015; pp. 153–181.
17. Hirsch, M.; Rodríguez, J.M.; Mateos, C.; Zunino, A. A two-phase energy-aware scheduling approach for cpu-intensive jobs in mobile grids. *J. Grid Comput.* **2017**, *15*, 55–80. [CrossRef]
18. Wagner, D.T.; Rice, A.; Beresford, A.R. Device Analyzer: Large-scale mobile data collection. *ACM SIGMETRICS Perform. Eval. Rev.* **2014**, *41*, 53–56. [CrossRef]
19. Kang, J.M.; Seo, S.S.; Hong, J.W.K. Personalized battery lifetime prediction for mobile devices based on usage patterns. *J. Comput. Sci. Eng.* **2011**, *5*, 338–345. [CrossRef]
20. Longo, M.; Mateos, C.; Zunino, A. A Model for Hour-Wise Prediction of Mobile Device Energy Availability. In *Information Technology-New Generations*; Springer: Berlin, Germany, 2018; pp. 351–358.
21. Singh, K.V.; Raza, Z. A quantum-inspired binary gravitational search algorithm–based job-scheduling model for mobile computational grid. *Concurr. Comput. Pract. Exp.* **2017**, *29*, e4103. [CrossRef]
22. Shah, S.C. Energy efficient and robust allocation of interdependent tasks on mobile ad hoc computational grid. *Concurr. Comput. Pract. Exp.* **2015**, *27*, 1226–1254. [CrossRef]
23. Lee, J.; Choi, S.; Gil, J.; Suh, T.; Yu, H. A scheduling algorithm with dynamic properties in mobile grid. *Front. Comput. Sci.* **2014**, *8*, 847–857. [CrossRef]
24. Kotz, D.; Henderson, T. Crawdad: A community resource for archiving wireless data at dartmouth. *IEEE Pervasive Comput.* **2005**, *4*, 12–14. [CrossRef]
25. Rodriguez, J.M.; Mateos, C.; Zunino, A. Energy-efficient job stealing for CPU-intensive processing in mobile devices. *Computing* **2014**, *96*, 87–117. [CrossRef]
26. Hirsch, M.; Rodriguez, J.M.; Zunino, A.; Mateos, C. Battery-aware centralized schedulers for CPU-bound jobs in mobile Grids. *Pervasive Mob. Comput.* **2016**, *29*, 73–94. [CrossRef]
27. Furthmüller, J.; Waldhorst, O.P. Energy-aware resource sharing with mobile devices. *Comput. Netw.* **2012**, *56*, 1920–1934. [CrossRef]
28. Guyon, I.; Elisseeff, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
29. Guyon, I.; Elisseeff, A. An introduction to feature extraction. In *Feature Extraction*; Springer: Berlin, Germany, 2006; pp. 1–25.
30. Hoerl, A.E.; Kennard, R.W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* **1970**, *12*, 55–67. [CrossRef]
31. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
32. Sak, H.; Senior, A.; Beaufays, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In Proceedings of the Fifteenth Annual Conference of the International Speech Communication Association, Singapore, 14–18 September 2014.