Rapid Energy Consumption Pattern Detection with In-Memory Technology

Christian Schwarz,* Felix Leupold,* Tobias Schubotz,* Tim Januschowski,† and Hasso Plattner*
* Hasso Plattner Institute, University of Potsdam
Potsdam, Germany
Email: {firstname.lastname}@hpi.uni-potsdam.de
† SAP Innovation Center
Potsdam, Germany
Email: tim.januschowski@sap.com

***Abstract*—The transformation of today's energy market poses new challenges for both, energy providers and customers alike as the usage of renewable energy sources and energy-awareness increases. Additionally, the energy infrastructure is changing fundamentally. On the one hand, the installation of so called smart meters offers the possibility of more detailed monitoring and fine grained electricity billing. On the other hand, the amount of data produced within the power grid increases dramatically. Utility companies will use such data to increase prediction accuracy and to improve energy production, while consumers will more and more transform to prosumers. Within that environment the necessity of short-term predictions increases to improve the power grids stability. In this article, we respond to some of the challenges that energy consumers and providers face by an implementation of a prototypical recording, monitoring and analysis landscape that uses smart meter data. The challenges that this article tackles include: real-time energy consumption classification; mass energy consumption data classification; and early short-term energy consumption prediction. In extensive experiments on real-world data, we show that such challenges can be handled effectively. We leverage smart meter data via a novel combination of machine-learning algorithms and latest in-memory technology.**

Keywords- energy pattern recognition; smart meters; machine learning; in-memory database; in-memory technology; interquartile range coverage

I. Introduction

The energy sector is currently undergoing transformational changes: Energy providers are facing new challenges and energy consumers are increasingly aware of their consumption and the associated costs.

For energy providers, the rise of renewable energy sources, such as solar or wind energy, drives the evolution from a purely consumption controlled supply network to a production controlled grid [2]. As the ratio of such energy sources increases [3], energy providers must now, more than ever, predict future energy consumption by their consumers earlier on the most up-to-date data in order to match their energy supply with their customers' demand. If energy supply is not sufficient for the predicted demand, the energy provider may decide to buy missing resources from other providers or provide incentives for the customer to change their behavior [4], e.g., via dynamic pricing. In every case, early and accurate energy consumption is essential for the energy provider.

For energy consumers, high environmental awareness [5] as well as economical necessities enforced by rising energy prices [6] drive conscious choices for energy consumption. In the industrial sector, energy expenses can make up to 43% of all operational expenses [7]. With ever increasing energy prices, it is essential for companies to control their spending on energy. Therefore, many companies monitor their energy usage on a more detailed level than most private customers. Companies have successfully reduced their energy consumption, for example, by 58% in the Aluminum industry since 1975 [7]. In the private household sector, a study by the US department for energy showed that simple monitoring energy consumption on in-home displays leads to different consumer behavior [8]. The study showed that 71% of private households changed their energy usage behavior – even if the initial savings only range from 4 to 15% [9]–[11], reported to stagnate at 7.8% on the medium-term [12]. With the increasing number of installed smart meters, private households are expected to monitor their energy consumption on a more frequent basis, leading to an increasing amount of computing power to satisfy the consumers' service needs. Summarizing, it is essential for both energy consumer groups to better understand their own energy consumption behavior.

This article considers pattern detection on energy consumption data. The deployment of pattern detection has the potential to help both, energy providers and energy consumers in their adaptation to the changing energy landscape. Providers can use energy consumption pattern classification to predict upcoming energy usage early on. Pattern detection is particularly useful towards such goals because very often, energy usage is highly regular (e.g., the energy consumption pattern of a manufacturing device). Once the (partial) energy consumption pattern is classified, an already known energy usage pattern of the same class can be used for early energy usage prediction. Energy consumers can use pattern detection to classify their existing energy footprint. Such classification could directly result in savings, for example, by identifying which high-energy consuming patterns occur at high energy price times and subsequently move them to low price times.

Pattern detection on energy consumption data is a big

data challenge by its nature. Additionally, there is a particular focus on real-time data with energy consumption data. The reason for this is the Advanced Metering Infrastructure (AMI) [13] which provides large amounts of real-time data on energy consumption. Processing and storing this data is a challenge in itself – running complex analyses on such real-time data was infeasible with regards to the business value until recently. However, with state-of-the-art infrastructure, such as in-memory technology [14], analytics on large, real-time data is now possible. In-memory technology is therefore a perfect match for implementing pattern detection on energy consumption data.

In short, the contribution of this article is as follows. Using an implementation of in-memory technology [15] as a key component of a monitor, record and analyze environment for energy consumption data, the computational feasibility of energy consumption pattern classification for various use cases is assessed. Extensive experiments show that pattern detection is not only computationally feasible, but also fast and accurate. Classic machine-learning algorithms as well as a purpose-built algorithm (so-called Inter Quartile Range Coverage) for pattern detection are used. Our experiments use real-world energy consumption traces, which were collected for this purpose and published jointly with this article.

This article extends our previous work [1]. Large sections of the original paper were re-written for improved exposition. Additional detail on the data of the experiments is given and the consumption data used in the experiments is published [16]. The main change consists of a more comprehensive experimental evaluation.

The remainder of this article is essentially organized into two parts. In the first part, the foundation for the extensive experimental evaluation that makes up the second part is considered. When considering the foundations for the experiments, Section III presents the used pattern classification methods. Section IV describes the data on which the experiments are conducted. The data collection process by our monitor, record and analyze infrastructure is described and we comment on some characteristics of the data. In the second part consisting of Section V, the pattern recognition algorithms are evaluated in a series of experiments. The conclusion and an outlook on future work are presented in Section VI. We begin with discussing related work and technological foundations next.

II. Background: Related Work and Technological Foundations

In this section, we discuss the background of this article, in particular work related to our research and we describe the technological and infrastructural aspects leading to our decision of using components of in-memory technology as the technological foundation for our work.

A. Related Work

The presentation of work related to this article is structured along the information flow in our experimental set-up, i.e., along the three steps monitoring, recording and analysis. First, energy consumption monitoring and then energy consumption recording infrastructure is discussed, followed by reports on literature on the usage of such information via advanced analytics on such data.

For the considered scenario, smart meters and, more generally speaking, smart grids are fundamental. They are considered to be the continuation of the classical power grid in the information age [4]. In order to avoid different conflicting standards amongst its participant countries, the European Union has instantiated the Smart Meter Coordination Group (SMCG) [17]. In this context, the OPENmeter project has proposed the AMI to be used for the smart grid [18]. Additionally, the Open Metering System has proposed a standard for communication between metering utilities that is independent of the metering devices' manufacturer. Open Metering Systems collaborates with SMCG and also assumes the AMI [17], [19]. Our experimental architecture to monitor and collect energy consumption data in Section IV is similar to the AMI.

Smart meter data will typically be stored in some database. For a number of years, in-memory databases and in particular the so-called in-memory technology, which makes heavy use of column-oriented in-memory databases, have received considerable attention in the literature [1], [14], [20], [21]. In-memory technology can be used to access smart meter data quickly, even though formerly, a column-oriented data layout was perceived not to be well suited for write-intensive workloads originating from a smart grid. To enable the handling of write-intensive workloads as they occur within an AMI, column-oriented in-memory databases use techniques like write-optimized differential buffers and bulk loading [20], [22]. Such techniques have proved that column-oriented in-memory databases are also useful in write-intensive workloads. Apart from storing and providing access to data, in-memory databases have been shown to provide additional benefits: a recent trend in the literature is the usage of in-memory technology for advanced analytical scenarios [4], [23], [24]. The present article extends this line of work.

Once stored in a database, collected smart meter data can be used for various use cases. Optimizing consumer contracts [25] and charging [4] are examples.

Another use case is prediction of energy consumption. Predicting the energy consumption for medium and shorter terms has been done for example with SVMs, e.g., [26], and artificial intelligence approaches such as neural networks [27], [28]. Further related work has focussed on comparing different algorithms for efficient pattern matching over event streams [29]. The general literature on machine learning algorithms is very rich. SVMs are discussed, for example, in [30], [31]. Duan and Keerthi discuss various multi-class implementations of SVMs [32]. Shakhnarovich et al. provide an overview of theory and application of the knn algorithm [33].

This article is set apart from existing approaches by two novel aspects. First, it considers algorithms that were previously not discussed in conjunction with in-memory technology. Second, a new and relevant use-case for in-memory technology is considered: energy consumption pattern detec-

tion. We are not aware of other approaches in the literature that combine machine learning algorithms with in-memory technology for energy consumption pattern detection.

B. Aspects of In-Memory Technology

Utility companies store their data within relational databases in so-called utility systems. The database as the single source of truth within the central system is a logical entry point for implementing new applications that work on real-time energy data like the pattern matching algorithms proposed in this article. In initial experiments, we found that conventional, row-oriented and disk-based database systems could not fulfill the interactive performance needs of our industry-scale sized scenario. This is due to the pure size of the collected data (471 million tuples in the database, including 27 million of the appliance used for the experiments). A disk-based column store like Sybase IQ [34], [35] or HP Vertica [36] would have presented a much better fit based on the analytical style queries needed for realising the scenario of interest of this article. These databases provide the usage of an relational model as presented in Section IV-A2, while reducing the required amount of storage space and hard disk seek times by using columnar storage layout and dictionary encoding. As an additional requirement, we want to run the database queries on real-time data with an ongoing stream of new data inputs into the database. This is the main reason for choosing SAP HANA [15] as the database engine because it provides storage space and combines analytical query optimized columnar layout with the insert performance of row-oriented, in-memory databases [20]. Despite the fact that SAP HANA is an in-memory database, data is stored persistently using database logging techniques, e.g., [37].

All algorithms rely on in-memory technology, in particular the in-memory database engine, for managing the in- and output of data. While all collected smart meter readings are stored within the database, the presented algorithms only need to operate on a pre-aggregated subset of the data. They compare the time series energy consumption data against a global repository of energy patterns. Therefore, the database has to select the corresponding data sets of a certain smart meter, aggregate the data onto a certain granularity level and present the data to the algorithm. Naturally, this step is included in each execution of the algorithms.

III. Pattern Recognition Algorithms

The field of pattern recognition studies the automatic discovery of similarities in data by using machine learning techniques. Pattern matching can be used for regression analysis and classification [30]. Regression analysis fits a function on a set of data points with the goal to extrapolate this data set. The task in classification is to decide whether a data point belongs to a certain class (or not). In this article, focus is on classification.

Supervised learning methods are used in the following. Contrary to un-supervised learning methods, a (mostly) correctly classified training set of data points is given [38]. The goal is to match time series consisting of energy consumption points to energy consumption classes. A classifier gets trained on the training data and is later on used to to classify time series that are not contained in the training set.

More formally, for an input vector $\vec{x}$, a classifier y that correctly classifies $\vec{x}$ into its class $\mathcal{C}$ is build. The existence of K classes is assumed. Supervised learning uses a learning phase, where it is given a set of training vectors X with the corresponding classes. The goal of the learning phase is to construct a (hopefully robust) classifier y that minimizes the classification error on the training set.

We selected the following three implemented pattern matching algorithms for a more detailed presentation: Inter-Quartile Range Coverage, a multi class support vector machine, and a k-nearest neighbor algorithm.

A. Inter-Quartile Range Coverage (IQRC)

The IQRC pattern matching algorithm was specifically implemented for our scenario to classify recorded patterns. Given a set of training vectors X^k with $|X^k| = n$ that belong to the same class $\mathcal{C}_k$, the upper and lower quartile for each component of each $\vec{x} \in X^k$, $\vec{x} \in \mathbb{R}^d$ are calculated. For simplicity of notation, n is assumed to be an even number divisible by 4. The range between the upper and lower quartile is called *inter-quartile range (IQR)*. For each class, d IQRs based on the training vectors are calculated, one for each component. More formally, let us define $\vec{v}_i^k$ as a non-decreasingly ordered sequence containing all n values from the training vectors X^k for component i. The element j is denoted by writing $(\vec{v}_i^k)_j$. So, $(\vec{v}_i^k)_{n/2}$ is the median of the sequence$\vec{v}_i^k$. We define $\mathrm{IQR}(\vec{v}_i^k) = [Q_1(\vec{v}_i^k), Q_3(\vec{v}_i^k)]$, where $Q_1(\vec{v}_i^k) = (\vec{v}_i^k)_{n/4}$ and $Q_3(\vec{v}_i^k) = (\vec{v}_i^k)_{3n/4}$.

IQRC is used to classify data as follows. Given a vector $\vec{x}$, the number of components of $\vec{x}$ that lie in the IQRs for $\mathcal{C}_k$ for $k \in \{1, \ldots, K\}$ is computed. If a previously set threshold for class $\mathcal{C}_k$ is exceeded, i.e., a set number of components of $\vec{x}$ lies within the IQRs of a class $\mathcal{C}_k$, $\vec{x}$ is classified as belonging to $\mathcal{C}_k$. It is possible that IQRC decides that $\vec{x}$ may belong to more than one class. In order to break such ties, the class is chosen, in which the threshold has been exceeded the most.

For classes with a high deviation amongst its members, the IQRs will be larger than for classes with a small deviation. In order to account for this, the weight of a component i lying in the IQR of a class $\mathcal{C}_k$ is set as

$$\frac{1}{1 + Q_3(\vec{v}_i^k) - Q_1(\vec{v}_i^k)} \,. \tag{1}$$

Note that $Q_3(\vec{v}_i^k) - Q_1(\vec{v}_i^k)$ may equal 0 for certain components i. With the weight as defined in (1), we rate those components that lie in smaller IQRs higher than values that lie in a greater IQRs.

The classifier y for IQRC is then formalized as follows. Let $\delta(k)$ denote the threshold of class $\mathcal{C}_k$.

We have

$$y(\vec{x}) = \arg\max_{k \in K} \left(\left(\sum_{i \in I} w(\vec{x}_i, \vec{v}_i^k) \right) - \delta(k) \right)$$

$$\text{where } w(\vec{x}_i, \vec{v}_i^k) = \begin{cases} \frac{1}{1+Q_3(\vec{v}_i^k)-Q_1(\vec{v}_i^k)}, & \text{if } \vec{x}_i \in \text{IQR}(\vec{v}_i^k), \\ 0, & \text{else.} \end{cases}$$

Recall that if more than one class has an IQRC above the threshold, the class for which the threshold is exceeded the most gets chosen. For a relatively high overlap among the classes, it is challenging to identify a threshold that is exceeded by all positive but by none of the negative examples. In the training phase of our algorithm, therefore $\delta(k)$ must be chosen carefully for $k \in K$. We propose a modified hill climbing algorithm [39] in the following. The optimization goal is to maximize the number of true positives, while false positives should be minimized.

Initially, the threshold for each class is determined such that none of the training patterns are classified. Furthermore, the classes are ordered by the size of the corresponding training sets non-increasingly. We start to decrement the first threshold as long as the number of correctly classified training vectors increases. If no further increase occurs, this threshold is then fixed for the class. We continue with the next class to decrement the threshold as long as the number of correctly classified training vectors increases. This step is repeated until all classes were considered. After processing all classes, a new iteration over the classes is started. Contrary to the first iteration, the thresholds of all other products can be assumed to be at a reasonably good value. The iterations over all products are continued until, for one entire iteration over all products, the number of correctly classified vectors does not improve. In our scenario, this typically happens after four iterations.

To place less burden on the choice of the threshold for correct classification, it would also be interesting to consider another variant of IQRC where proximity to the median is weighted additionally. We leave this idea for future work.

B. Multi-Class Support Vector Machine

We also consider classification by Support Vector Machines (SVMs) [31]. SVMs offer binary classification. In our scenario, we aim at classification into K classes, with $K > 2$ typically. The most common approach to extend SVMs for such multi-class classification is the *one-versus-all* approach [32] which we refer to as MCSVM. In the training phase where we have K classes $\mathcal{C}_1, \ldots, \mathcal{C}_n$ and corresponding training vectors, we create K binary SVMs, one for each class. The SVM corresponding to class $\mathcal{C}_i$ is trained with all training vectors from C_i for its first target and with the rest of the training vectors for the other target.

When classifying a pattern that is not contained in the training set, this incoming pattern is passed to each SVM. Ideally, only one SVM detects a positive result. If there is more than one SVM classifying the input as $\mathcal{C}_i$, then the one with the largest result vector is used. If there is no SVM classifying the input as $\mathcal{C}_i$, the one with the smallest negative result vector is chosen. Assuming K classes, this approach needs to test K SVMs.

C. K-Nearest Neighbor

When looking at our energy consumption data, we noticed that the energy consumption patterns have a considerable variance, even if they belong to the same class. Clustering energy consumption patterns into their corresponding classes leads to rather big and possibly even overlapping clusters. Therefore, we also consider classifying an energy consumption pattern by looking for the pattern that is most closely related to the pattern to be classified. The intuition for this is as follows. In a subspace with many energy consumption patterns of class $\mathcal{C}_1$, a pattern of class $\mathcal{C}_2$ varying from the others might still be identified as one that more closely resembles the input and should therefore be chosen. This is what the k-nearest neighbor algorithm does. In the following, we refer to this algorithm as the *knn* algorithm. In our case, it suffices to set k to 1. Given an input vector to be classified, the knn classifier returns the class of the training set element for which the distance is minimal [33]. For simplicity and the fact that our vectors represent continuous variables, we use the Euclidian distance as metric.

An advantage of the knn algorithm is the potential for speed-up by parallelization. Also, there is no computationally expensive learning phase required.

IV. Smart Meter Data

The evaluation of the algorithms described in Section III requires an appropriate data set. While some smart meter data is publicly available, e.g., [40], such data sets are typically too small to be useful in experiments on industry-scale data, which are the focus of this article. Since we also did not want to rely on artificially generated data for our experiments, we decided to record smart metering energy consumption data ourselves. In this section, we describe the experimental set-up for collecting the energy consumption data. We also describe some characteristics of the energy consumption data and the used data model. We decided to make the energy consumption of the used appliance data publicly available [16], in order to, hopefully, facilitate future research in the area of energy pattern classification.

A. Data Collection

Recording real-world energy consumption data presents a challenge in itself [19]. We responded to this challenge by setting up the following experimental environment. We monitored all electric energy consuming devices inside a shared space of our research group and recorded their energy consumption over a period of three months.

1) Monitoring: The electric devices that we monitored represent a subset of the major devices in households: television and home entertainment components, regular illumination, IT components, two fridges and a coffee machine. In our experiment, we measure all devices independently.

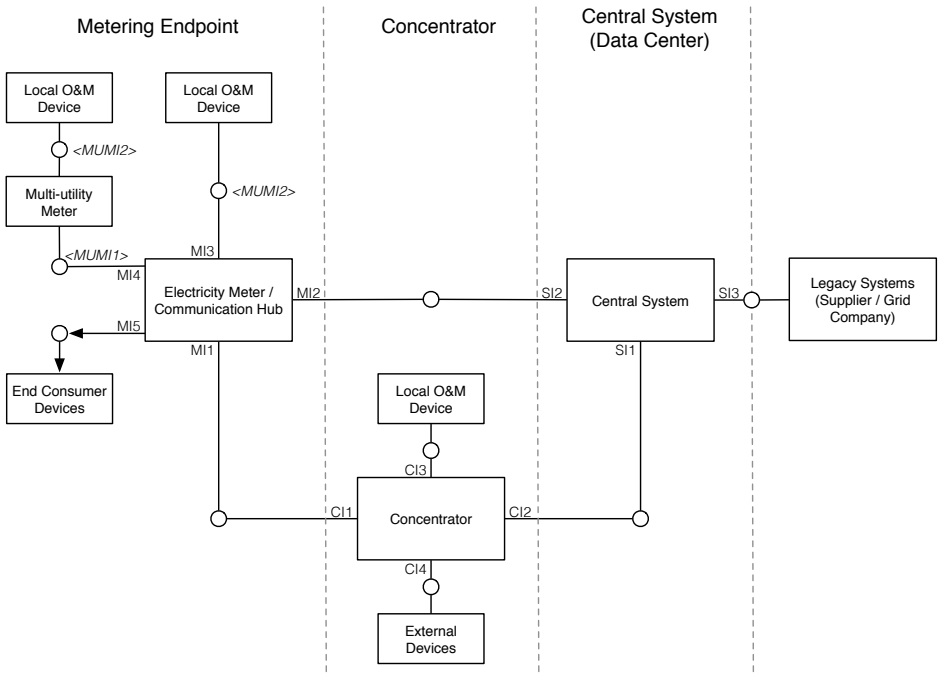


Figure 1. FMC Illustration of Advanced Metering Infrastructure, as defined in [13].

With the costs of self-measuring devices in mind, we decided to use an Emerson Network Power Rack Power Distribution Unit (PDU) with a Liebert MPX control module [41] for monitoring the energy consumption. This PDU is capable of measuring energy data for each connected device independently, e.g., power, voltage, current and rating. The measuring accuracy of the PDU is $\pm 10\%$. Throughout our experiments, we will use the consumption data of the coffee machine for pattern detection purposes. From our collected data, we chose the coffee machine because it presents the toughest challenges for our algorithms. For example, energy consumption varies depending on the coffee being made whereas the energy consumption of a fridge does not vary as much. Both, private households and industry have started to adapt smart meters in order to monitor energy consumption [42]. We expect smart meter to become more prevalent in the future.

2) Recording: For the recording of the energy consumption of the devices, we created an AMI-like multi-level architecture [19], where the energy consumption readings of devices are transmitted via a concentrator component to a central system. In our case, we chose an in-memory database as this central system. This architecture closely resembles the AMI that is expected to be applied for collecting energy consumption data within the power grid of the future. We depict the AMI in Figure 1.

Figure 2 contains a schematic view of our recording infrastructure. The PDU itself is connected to a local area network and data is retrieved using the Simple Network Management Protocol.

The data collector queries the PDU in average once per second to collect the data for each device. Based on these physical measurements, we calculate the power consumption. Finally, we transfer the energy consumption data into the in-memory database. The resulting transmission interval from PDU to database is between 0.5 and 2 seconds depending on the current traffic on our local Ethernet network.

Table I
SCHEMA OF THE TABLES USED FOR PREDICTION.

DEVICE_READINGS	
DEVICE_ID	INTEGER
DATETIME	INTEGER
CONSUMPTION	FLOAT

PATTERN_RECOGNITION	
DATETIME	INTEGER
CONSUMPTION	FLOAT
PRODUCT	FLOAT

We store the collected data in a table called `device_readings`. The occurrence of a pattern is recorded in `pattern_recognition`. The entire database schema is depicted in Table I.

B. Training Set

As mentioned in Section III, supervised machine learning techniques are used for energy pattern detection in our work. Therefore a set of correctly classified energy consumption pattern is needed that can be used to train the algorithms.

The classification challenge for the coffee machine is to detect the type of product that the coffee machine produces, e.g., cappuccino, hot milk or espresso, based on the energy consumption. During the beginning of our data monitoring

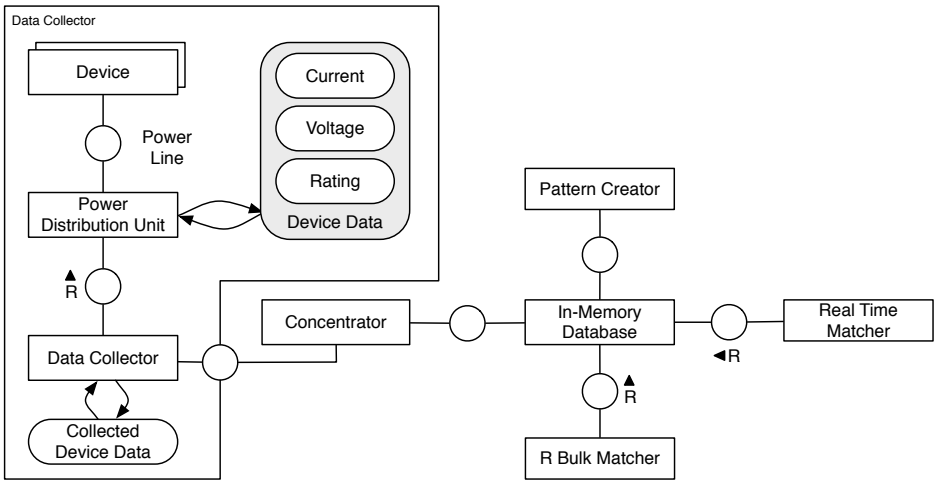


Figure 2. Experimental setup as FMC block diagram.

and recording phase, users of the coffee machine were asked to input the type of coffee product they had selected into a purpose-built application next to the coffee machine. This classification is stored together with the energy consumption data and thus creates a training set for our supervised machine learning algorithms.

Before using this data as a training set for the algorithms, a simple data cleansing algorithm is performed to eliminate energy consumption patterns that have more than three times the standard deviation from the other patterns in their class [43]. Tables II summarizes the training set.

Table II
DETAILS ON THE TRAINING SET.

Product	Number of Occurrences	%
cappuccino	9	4
cleaning	33	15
single espresso	10	4
double espresso	13	6
single coffe	58	26
double coffee	7	3
latte macchiato	89	39
only milk	7	3
total	226	100

Figure 3 shows quartiles of two selected energy patterns, latte macchiato and single coffee. We see the difference between both patterns clearly. However, we note that there is also a large spread of values in each pattern. A number of factors may cause this. For example, measuring inaccuracies by the PDU or coffee-machine inherent reasons such as the coffee water having varying temperatures. It is this large spread among the patterns which makes classification such a challenge.

C. Publicly Available Consumption Data

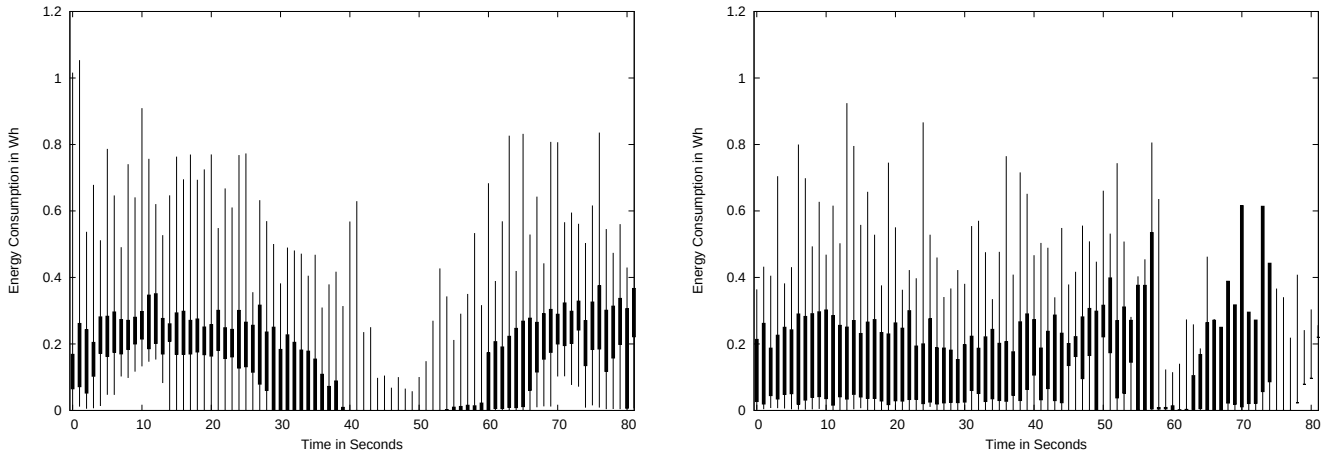
A condensed energy consumption data of the coffee machine used in our experiments is now publicly available [16]. We removed most records that correspond to an idle state of the coffee machine. Table III contains details for the database schema of [16]. Apart from the energy consumption details, the data also contains the classification, whenever it is available. Even though our data comes from a comparably narrow scenario, we think that more general conclusions based on this data are possible. First, the data consists of real, noisy energy consumption. Therefore, it is more appropriate than randomly generated data. Second, the energy consumption traces present a considerable variety of patterns as we noted in Section IV-B. Third, and most important, the sheer size of the data allows for direct conclusion on the computational feasibility of energy pattern detection use cases. The original data set of the coffee machine consists of roughly 27 million tuples, which correspond to one month history of current smart meters for roughly 10,000 households.

Table III
SCHEMA OF PUBLISHED ENERGY DATA.

DEVICE_READINGS		
UNIXTIME	FLOAT	Time in seconds from 01.01.1972
DEVICE_ID	INTEGER	ID of the device at the PDU
POWER_STATE	INTEGER	Power state of the output (2=on)
POWER	INTEGER	Power at the PDU output in W
VOLTAGE	FLOAT	Voltage at the PDU output in V
AMPERAGE	FLOAT	Amperage at the PDU output in A
PATTERN	INTEGER	Identified pattern

V. EVALUATION

In this section, we evaluate the algorithms from Section III on the data presented in Section IV. In our experiments, we evaluate in-memory technology for different energy consumption pattern detection use-cases such as: computational feasibility of on-line pattern classification, accuracy of classification and short-term prediction. Before going into details of the experiments, we give further details on the experimental set-up.



(a) Latte Macchiato. (b) Single Coffee.

Figure 3. Example Quartile Plots for Coffee Products.

A. Preliminaries

For our experiments, we use a HP ProLiant DL580 G7 series server that is equipped with four Intel Nehalem X7560 CPUs and 256 GB main memory. The server runs a 64-bit version of openSUSE 11.2 (kernel 2.6.31.14). We use an instance of SAP's implementation of in-memory technology called HANA [15]. As mentioned in Section II-B, among HANA's features is a column-oriented data layout that is particularly well suited for analytical workloads. Our implementation uses the System R [44] interface of the database development version. The tight integration of R into HANA is a further reason for choosing in-memory technology. Our implementations of knn and MCSVM rely on the rminer package [45].

1) Classification Accuracy Benchmarking: When we benchmark the classification accuracy of our algorithms, we divide our training data set into two parts. One set is used for training the algorithms, while the other set is required for testing the accuracy by comparing the output classification of the algorithms with the actual classification.

We use a cross validation technique to achieve reliable results. The pattern set of each class/product is split into five parts. Each iteration of the algorithm, uses four parts for training and one for testing purposes. The overall performance is calculated by taking the average over all iterations. This technique is called *leave-some-out cross validation* [46].

2) Data Features Used for Classification: In order to classify energy consumption patterns, we use a set of features of each pattern to run our classification algorithms on. Complementing the raw data for electronic power consumption in watt hours, we additionally consider the following features of the gathered data. Let $\vec{x} \in \mathbb{R}^d$ be an energy-consumption pattern.

- Number of peaks: the number of local maxima in $\vec{x}$, i.e., $|\{x_j : x_{j-1} < x_j \wedge x_j > x_{j+1}, 1 < j < d\}|$.
- Greatest Delta: $\max_{i=1,\ldots,d} x_i - \min_{i=1,\ldots,d} x_i$
- Sum: $\sum_{1 \leq i \leq d} x_i$
- Duration: d
- Moving Average: a time series $\vec{a} \in \mathbb{R}^{d-k}$ with $\vec{a}_i = 1/k \sum_{1 \leq j \leq k} x_{i-j/2}$ for appropriate i and suitably chosen k. Within our experiments, $k = 4$ produced the best results, reducing the effect of outliers on the local estimate without over-smoothing the time series.
- Histogram: a sequence of occurrences of distinct energy consumption values ordered non-decreasingly by energy consumption values.

B. Computational Performance of Real-Time Classification

In the first experiment, the computational feasibility of classifying energy consumption patterns is evaluated. Computational speed is important because of the following reasons. Fast response times of our classifiers are mandatory to enable close to real-time matching. The faster the response time is, the earlier the short-term demand can be predicted. This may have direct monetary consequences. Furthermore, only a fast classification allows interaction for which the response limit is 2 seconds [47]. Therefore, queries that are triggered by real-time classification have to be answered within this time interval.

Real-time classification is implemented as a background process that performs classification cycles periodically. When the coffee machine is idle, one cycle takes about three milliseconds. If the coffee machine is not idle, i.e., is currently producing, a cycle still takes less than a second. Table IV shows the cycle times for the different algorithms. It can be seen that knn is the fastest algorithm, on average as well as in the worst case. Matching with MCSVM takes about 30% longer. IQRC needs about twice the time compared to knn. Overall, our experiments clearly show that all algorithms have satisfying performance, as they are well below the critical limit of 2 seconds.

C. Accuracy of Pattern Classification

Our next experiment tests the ability of our algorithms to classify energy consumption patterns after they have been

Table IV
COMPUTING TIMES FOR OUR ALGORITHMS FOR REAL-TIME CLASSIFICATION: AVERAGE AND EXTREME COMPUTING TIMES.

Algorithm	Shortest time in ms	Longest time in ms	Average time in ms
knn	3	806	9
MCSVM	3	1116	10
IQRC	3	1547	13

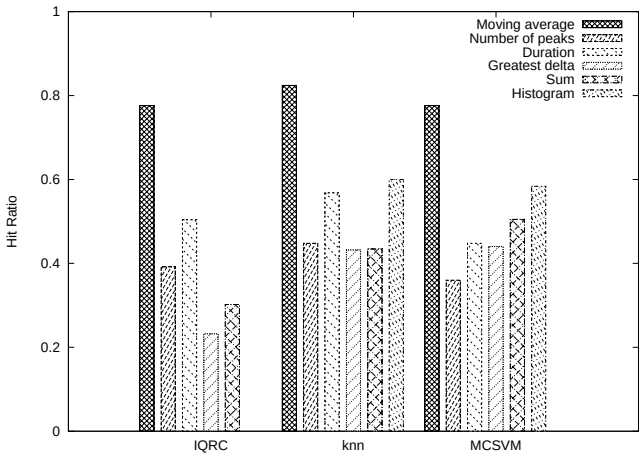


Figure 4. The comparison of three algorithms with different data features.

observed in full. Figure 4 shows the ratios of correctly classified patterns over all patterns for all data features from Section V-A2. We refer to this ratio as the *hit ratio*. We test IQRC, knn, and MCSVM with these features. We remind the reader that we use these features additionally to the raw energy consumption data, i.e., the energy consumption time series themselves.

For IQRC, we were not able to perform the benchmark with the histogram feature, because all patterns of one product result in one histogram. They do not have any deviations or quartiles. The lower boundary for the matching performance with eight different products is 12.5% which would be the accuracy of chance. Recall from Section IV that the PDU, which was used for measuring the consumption data, has an accuracy of $\pm 10\%$ for its measurements [41].

As we can see in Figure 4, the composite feature moving average performs almost equally well across all algorithms and outperforms by far all other features. We think that the smoothing achieved by moving average evens out un-natural deviations caused by measuring inaccuracies and therefore is closer to an idealized pattern.

In general, we observe that the richer the feature is, the more information can be used for classification. This results in higher hit ratios. Comparably simple features perform significantly worse than richer ones. They even seem to disturb the classification. Note that the histogram feature could only be implemented for knn and MCSVM. Though the histogram leads to a slightly better classification than other features, it is still noticeably worse than composite features. The reason is that the measured values hardly differ in size, because the histogram has a lot of different values with low frequencies.

If we compare the algorithms against each other, we notice that knn performs slightly better than the other two. It performs about 5 to 10% better than the other algorithms in all features except for the greatest delta and sum feature. For moving average, the IQRC algorithm has the same hit ratio as MCSVM. Nonetheless, IQRC outperforms MCSVM slightly considering the number of peaks and duration features. MCSVM on the other hand has the strongest results for the greatest delta and sum criteria. The implementation of MCSVM using *one-versus-all* is susceptible to mis-classification if all machines calculate a negative result [48]. Due to the high deviation amongst patterns in our scenario, this case occurs more often. Therefore, the overall accuracy of MCSVM is not as we initially had hoped for. However, due to the high deviation among energy consumption patterns, the hit ratio seems to be overall satisfactory.

D. Computational Performance of Bulk Pattern Recognition

Next, we analyze the computational feasibility with respect to computing times for bulk pattern recognition, where we compare the complete history of the energy consumption data with a set of defined patterns. This scenario is interesting for both the industrial and the private sector, because one could gain an in-depth understanding of the underlying mechanisms of energy consumption behavior. Having classified the energy consumption history allows analyses such as: how much energy was spent on which product/device, or which devices are primarily used during times of highest energy prices. Similar to the experiment in Section V-B, computing times are a critical factor to allow human interaction. However, due to the much larger amount of historical data (versus the live data in Section V-B), computing times will necessarily be significantly higher.

In our experiment, we calculated the average time for one cycle in the algorithm over multiple hours of operation. When we repeat the measurement for the different algorithms, we measure the same time slots on different days. We define one cycle as querying the database for new data plus the time used for matching given there is a pattern detected. Note that we also use energy consumption data that is not contained in the training set for this experiment. Due to the large amount of data involved, the use of an in-memory database is mandatory to allow reasonable computation times. For our experiments, accessing the largest set of data takes a few seconds.

Figure 5 shows the execution times of the MCSVM algorithm, depending on the number of readings in the `device_readings` table for different numbers of used cores (we shall comment on the number of cores further below). We chose this algorithm for our experiment because the MCSVM's computational performance is roughly an average of the other algorithms as the experiment in Section V-B showed. The values in Figure 5 represent the averages of ten measurements with a standard deviation of 11%. One reason for the comparatively large deviation in computing times comes from the fact that during our experiments, energy

consumption data was still being loaded into the database system. We did not stop the loading in order to guarantee more realistic experimental settings.

This experiment shows that the computation time for bulk matching grows linearly in the number of records. Note that both axes have a log-scale. This is particularly pronounced for more than 1000 records. Based on this and the low total computing times, we conclude that bulk matching is computationally feasible for data set sizes that match smart grid use cases as we commented in Section IV-C.

In Table V, we give the results of the bulk pattern classification. We do not give statistics on the accuracy, as we have already commented on this in Section V-C and we cannot measure accuracy on unclassified data. However, we note that the distribution of patterns resembles the training set in Table II.

In this experiment, we laid a special focus on parallelization. Note that the *rminer* package, on which the implementation of our algorithms is based, does not parallelize its computation. However, we parallelized the execution of our algorithms by partitioning the data. Each of the CPUs could then independently work on an equally sized fraction of the total values in the energy consumption data.

We remark that, independent of the degree of parallelization achieved (measured in the number of used cores in Figure 5), the computation times grows linearly with the number of records to be classified. This is expected. Also expected is a decrease of total running time for a fixed number of records with an increasing degree of parallelization. Somewhat unexpected is that this speed-up is comparably small. We explain this as follows. According to Ahmdal's law the speedup is determined by the serial fraction of the algorithm [49]. In our case, this fraction is determined by the initialization of the classifier and the merging of different results for partitions of the `device_readings` table. Merging these results for a total of one million data sets already takes 10 to 20 ms. Although we tried to parallelize this merge, the increase from eight to 32 processes even increases the execution time for less than 200,000 values. The overhead in the merge is not outrun by the smaller number of device readings which each process has to analyze. Nevertheless, 32 cores still outperform eight cores for more than 200,000 readings. With an increasing number of readings, we expect the gain from executing the computing expensive operations in parallel to increase further.

Table V
RESULTING PATTERNS FROM BULK PATTERN RECOGNITION.

Product	Number of Occurences	%
cappucino	82	2
clean	409	7
single espresso	819	15
double espresso	491	9
single coffee	1719	31
double coffee	82	1
latte macchiato	1801	33
only milk	82	1
total	5485	100

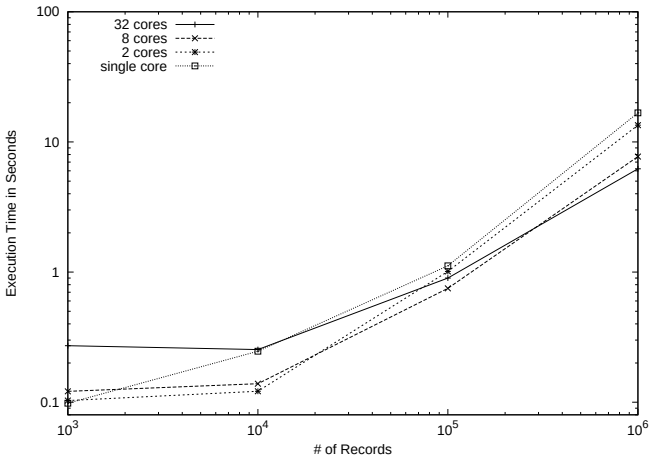


Figure 5. Comparison of execution times for bulk pattern recognition depending on the number of reading for different number of CPU cores.

E. Accuracy of Short-Term Energy Consumption Pattern Detection

In this experiment, we test the accuracy of classifying partial energy consumption patterns, i.e., classifying an energy consumption pattern before it is completed. The motivation for this experiment is as follows. If a pattern is detected, subsequent values of patterns from the same class can be used for predicting the future consumption of a device. The earlier we correctly classify the energy pattern, the more useful this classification becomes as the prediction period becomes longer. However, it is also more difficult to correctly classify patterns, the shorter they have been observed. This is because early classification has to be performed on incomplete energy consumption data and is therefore not as accurate as classification after the complete consumption. Therefore, we need to trade-off classification accuracy with point in time of classification.

Figure 6 shows the accuracy of the knn and MCSVM algorithm depending on the length of the patterns. If we pass a pattern with length n, we cut all training patterns down to that length and apply the algorithms.

We consider a classification rate of 0.5 to be sufficient in order to speak of successful pattern recognition. There are eight possible beverages, a success rate bigger than 50% would be four times better than chance. As we can see, we break the 0.5 accuracy line at approximately 20 seconds. This means that approximately one third of the pattern is sufficient for pattern recognition. If we transfer that finding to industrial manufacturing processes that take multiple hours, the moment of classification is early enough for utility companies, as it provides sufficient headroom for trading, e.g., at the EEX spot market [50].

Since we can classify the energy consumption after twenty seconds, we can predict the succeeding ten to seventy seconds using the information from our trained patterns. We consider this in the next experiment.

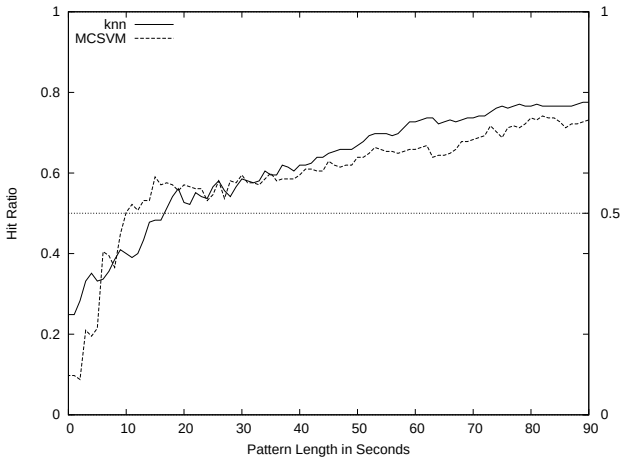


Figure 6. Hit ratio depending on the length of the input vector.

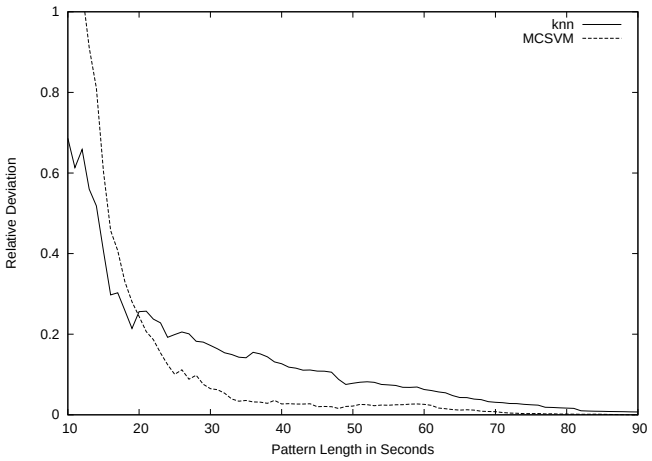


Figure 7. Accuracy of prediction over length of pattern.

F. Accuracy of Short-Term Energy Consumption Prediction

As a final use case of our pattern matching approach for energy consumption, we measure prediction accuracy of energy consumption. Apart from more obvious use cases where an energy provider could try to buy capacities based on her predictions, energy consumers could use predictions for anomaly detection. Whenever we have detected a pattern and its subsequent values differ considerably from the ones predicted, this could either mean that we have a pattern that we have not had in the training set or that something in the device causing the pattern went abnormally. In both cases, issuing a warning seems reasonable. Either the training set needs to be updated or the machine needs to be checked.

We implemented two methods for predicting energy consumption. The first method uses the knn algorithm from Section III-C as follows. We choose the most closely related pattern from our training set to the one having been partially observed. We use this energy consumption pattern from the training set as a prediction of future values for the current pattern. The second method uses the MCSVM method from Section III-B as follows. We start by classifying the observed pattern using our MCSVM method. Next, we identify a training pattern within this class with the least Euclidean distance to our observed pattern. We finally use the values of the training pattern for the prediction.

When we predict the subsequent consumption of a pattern, we have to balance prediction accuracy with time of prediction similar to Section V-E. It is clear that the longer we wait after a pattern has started, the more accurate we can predict the rest of the pattern. However, the longer we wait, the less valuable the prediction becomes. Managing this trade-off depends highly on the setting, e.g., on an economic cost model, and must be decided for the concrete use-case. This is shown in Figure 7. It shows the average prediction accuracy depending on the time the prediction is made. We measure accuracy as the absolute difference of the predicted consumption and the true consumption. This difference is divided by the true consumption.

Our motivation for using this error measure comes from the use case of short term energy forecasting by energy providers. These energy providers could use the total energy consumption of the short term forecast to decide how much energy they would need to provide in the next space of time.

Figure 7 shows that after 20 seconds, i.e., after less than one third of the pattern, we have an average deviation of 25% between prediction and actual consumption. For other prediction use cases, this value seems to be quite acceptable [23]. After 40 seconds, i.e., less than half of the duration of the pattern, the deviation falls even below 20%.

Considering that the consumption values of the coffee machine even under load ranges between 0.1 and 0.8 watt seconds, a predicted value that only differs by .01 watt seconds may lead to a deviation of 10%. Therefore we would have to predict three decimal places correctly to fall below that number. Recall that the accuracy of the PDU is only around $\pm 10\%$ which further complicates predictions. In more advanced scenarios, e.g., for high performance industrial machines, the consumption is higher than for the coffee machine. We expect the precision in measuring energy consumption for industrial use cases to be higher. This may lead to more accurate predictions because the training set may be better.

VI. Conclusion and Future Work

In this article, we presented a case study that suggests that leveraging real-world mass energy consumption data for smart analytics is computationally feasible. An essential component for the success of our case study is the deployment of in-memory technology as implemented in [15]. This in-memory database handles in-coming, live energy consumption data, while, at the same time, allowing analytics on the collected mass data with rapid response times.

As part of the contribution of this article, we make the energy consumption data that we used in our experiments public [16]. We think that the following general conclusions are possible based on the experimental evaluation on our data:

- Our experiments reveal that a number of use cases for energy consumption data analytics can be handled effectively with in-memory technology. Short-term prediction of energy consumption based on short-term classification of energy consumption traces into corresponding patterns is feasible. This opens up many opportunities both for energy providers and energy consumers. Energy providers could use short-term predictions for better trading and classification for better pricing. Energy consumers could use short-term predictions for early warning systems and classification for timing energy consumption better, for example, in order to reduce maximum energy consumption levels.
- While the length of the pattern is rather short in our data, our experiments suggest that after about 20 to 30% of any sufficiently distinct energy consumption trace, it may be recognized and predicted with sufficient accuracy independent of its absolute length. We believe that other energy consumption data sets could be easier to classify into different patterns since the patterns in our data are comparably similar.
- Classification of large real-world sized data sets is computationally feasible with in-memory technology. Such classification allows energy consumers and providers to deeply analyze and understand existing energy consumption data.

Future work may include evaluating the results on other data sets. For example, on energy consumption data from different types of manufacturing machines that produce different and more diverse energy usage footprints. A further possibility for future work would be unsupervised machine learning methods; in particular, benchmarking such unsupervised methods with the presented supervised methods in terms of accuracy and computational speed. Such unsupervised methods would provide useful insights in scenarios where no training data is available.

Finally, our experiments reveal that the speed of in-memory technology-based energy consumption pattern detection is such that machine-human interaction is possible, thus allowing a combination of human insight with machine learning algorithms. Enabling this human-machine interaction for energy consumption classification and prediction would be a most interesting avenue for future work.

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for their insightful comments that helped improve the article.

REFERENCES

[1] C. Schwarz, F. Leupold, and T. Schubotz, "Short-term energy pattern detection of manufacturing machines with in-memory databases – a case study," in *Proceedings of the Second International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies, ENERGY 2012*, 2012.

[2] P. Evans-Greenwood and A. Mulholland, "Moving the energy industry from demand- to supply-driven," http://www.slideshare.net/peg/moving-the-energy-industry-from-demand-to-supply-driven, 2007.

[3] eurostat, "Renewable energy statistics," http://epp.eurostat.ec.europa.eu/statistics_explained/index.php/Renewable_energy_statistics#Electricity, 2011.

[4] M.-P. Schapranow, R. Kühne, A. Zeier, and H. Plattner, "Enabling Real-Time Charging for Smart Grid Infrastructures using In-Memory Databases," in *IEEE 35th Conference on Local Computer Networks (LCN)*. IEEE, 2010, pp. 1040–1045.

[5] eurostat, "Environmental statistics and accounts in Europe," http://epp.eurostat.ec.europa.eu/cache/ITY_OFFPUB/KS-32-10-283/EN/KS-32-10-283-EN.PDF, 2010.

[6] Department of Energy & Climate Change, "Energy price statistics," http://www.decc.gov.uk/en/content/cms/statistics/energy_stats/prices/prices.aspx, 2012.

[7] The NEED Project, "Energy Consumption," *Intermediate Energy Infobook*, p. 46, 2011.

[8] L. A. Butler, "In-Home Display Pilot," *US Department of Energy - Energy Efficiency and Renewable Energy*, pp. 1–9, Jul. 2011.

[9] W. Abrahamse, L. Steg, C. Vlek, and T. Rothengatter, "A review of intervention studies aimed at household energy conservation," *Journal of Environmental Psychology*, vol. 25, no. 3, pp. 273–291, Sep. 2005.

[10] S. Darby, "The Effectiveness of Feedback on Energy Consumption," *Environmental Change Institute, University of Oxford*, pp. 1–24, Apr. 2006.

[11] ——, "Energy feedback in buildings: improving the infrastructure for demand reduction," *Building Research & Information*, vol. 36, no. 5, pp. 499–508, Oct. 2008.

[12] S. S. van Dam, C. A. Bakker, and J. D. M. van Hal, "Home energy monitors: impact over the medium-term," *Building Research & Information*, vol. 38, no. 5, pp. 458–469, Oct. 2010.

[13] The OPENmeter Consortium, "Report on the identification and specification of functional, technical, economical and general requirements of advanced multi-metering infrastructure, including security requirements," *Deliverables*, June 2009.

[14] H. Plattner, "A common database approach for OLTP and OLAP using an in-memory column database," *Proceedings of the 35th SIGMOD International Conference on Management of Data*, Jun. 2009.

[15] SAP AG, "SAP HANA product information," www.sap.com/HANA, 2012.

[16] C. Schwarz, F. Leupold, T. Schubotz, T. Januschowski, and H. Plattner, "Energy consumption data," http://inmemoryeffect.com/energy/data/, 2012.

[17] Arbeitsgemeinschaft Für Sparsame und Umweltfreundlichen Energieverbrauch E.V., "Smart Meter - Intelligente Zähler."

[18] OPENmeter, "Requirements of AMI," Tech. Rep., 2009.

[19] H. Baden and P. Gabriel, "Open metering system specification," Open Metering System Group, Tech. Rep., 2011.

[20] V. Sikka, F. Faerber, W. Lehner, S. K. Cha, T. Peh, and C. Bornhoevd, "Efficient transaction processing in SAP HANA database: the end of a column store myth," in *Proceedings of the 2012 SIGMOD International Conference on Management of Data*, May 2012, pp. 731–742.

[21] H. Plattner and A. Zeier, *In-Memory Data Management*. Springer, 2011.

[22] J. Krueger, M. Grund, C. Tinnefeld, and H. Plattner, "Optimizing write performance for read optimized databases," *Database Systems for Advanced Applications*, 2010.

[23] T. Januschowski, M. Lorenz, C. Schwarz, E. Folkerts, R. Heimburger, A. Akkas, N. Youssef, and D. Simchi-Levi, "Demand forecasting with partial POS data using in-memory technology," in *Proceedings of the 32nd Annual International Symposium on Forecasting*, 2012.

[24] J.-H. Boese, G. Rabinovitch, M. Steinbrecher, M. Magarian, M. Marcon, C. Tosun, and V. Sikka, "Data mining in life sciences using in-memory DBMSs: A case study on SAP's in-memory computing engine," in *Proceedings of Business Intelligence for the Real Time Enterprise*, 2012.

[25] C.-C. Chuang, J. Y. C. Wen, and R.-I. Chang, "Consumer Energy Management System: Contract Optimization using Forecasted Demand," in *ENERGY 2011: The First International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, 2011, pp. 58–63.

[26] B.-J. Chen, M.-W. Chang, and C.-J. Lin, "Load forecasting using support vector machines: a study on EUNITE competition 2001," *Power Systems, IEEE Transactions on*, vol. 19, no. 4, 2004.

[27] P. A. Gonzalez and J. M. Zamarreto, "Prediction of hourly energy consumption in buildings based on a feedback artificial neural network," *Energy and Buildings*, vol. 37, pp. 595 – 601, 2005.

[28] S. A. Kalogirou, "Applications of artificial neural-networks for energy systems," *Applied Energy*, vol. 67, no. 1–2, pp. 17–35, 2000.

[29] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman, "Efficient pattern matching over event streams," in *Proceedings of the 2008 SIGMOD International Conference on Management of Data*, 2008, pp. 147–160.

[30] S. Marsland, *Machine Learning: An Algorithmic Perspective*. Chapmann & Hall/CRC, 2009.

[31] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, pp. 273–297, 1995.

[32] K. Duan and S. S. Keerthi, "Which is the best multiclass SVM method? An empirical study," in *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, 2005, pp. 278–285.

[33] G. Shakhnarovich, T. Darrell, and P. Indyk, Eds., *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. The MIT Press, 2006.

[34] C. D. French, "One size fits all database architectures do not work for dss," 1995.

[35] Sybase, Inc., "SAP Sybase IQ Columnar Database," http://www.sybase.com/products/datawarehousing/sybaseiq, 2012.

[36] Vertica, "Columnar storage and execution," http://www.vertica.com/the-analytics-platform/columnar-storage-execution/, 2012.

[37] J. Lee, K. Kim, and S. K. Cha, "Differential logging: A commutative and associative logging scheme for highly parallel main memory database," 2001, pp. 173–182.

[38] C. M. Bishop, *Pattern Recognition And Machine Learning*. Springer, 2007.

[39] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (2nd Edition)*, ser. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2002.

[40] OpenEI, "Energy datasets," http://en.openei.org/datasets/, 2012.

[41] Emerson Electric Co., "User Manual – Network Interface card for the Liebert Rack PDU family of power distribution products," Monitoring For Business-Critical Continuity, Tech. Rep., 2009.

[42] U. E. Information, "How many smart meters are installed in the U.S. and who has them?" http://www.eia.gov/tools/faqs/faq.cfm?id=108&t=3, 2010.

[43] D. Ruan, G. Chen, E. E. Kerre, and G. Wets, Eds., *Intelligent Data Mining: Techniques and Applications*, ser. Studies in Computational Intelligence. Springer, 2005.

[44] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2011.

[45] P. Cortez, "Data Mining with Neural Networks and Support Vector Machines using the R/rminer Tool," in *Advances in Data Mining – Applications and Theoretical Aspects, 10th Industrial Conference on Data Mining*. Berlin, Germany: LNAI 6171, Springer, 2010, pp. 572–583.

[46] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1995, pp. 1137–1143.

[47] W. O. Galitz, *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. Wiley & Sons, 2007.

[48] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, pp. 415–425, 2002.

[49] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," *Proceedings of the 30th AFIPS 1967 Spring Joint Computer Conference*, pp. 483–485, 1967.

[50] European Energy Exchange AG, "Transparency in energy markets," Tech. Rep., 2011.