Open Access

Open Journal of Big Data (OJBD)
Volume 4, Issue 1, 2018
http://www.ronpub.com/ojbd
ISSN 2365-029X

Modelling Patterns in Continuous Streams of Data

Ricardo Jesus[A, B], Mário Antunes[A, B], Diogo Gomes[A, B], Rui L. Aguiar[A, B]

[A] DETI, Universidade de Aveiro, 3810-193 Aveiro, Portugal, ricardojesus@ua.pt
[B] Instituto de Telecomunicações, Universidade de Aveiro, 3810-164 Aveiro, Portugal, {mario.antunes, dgomes, ruilaa}@av.it.pt

Abstract

The untapped source of information, extracted from the increasing number of sensors, can be explored to improve and optimize several systems. Yet, hand in hand with this growth goes the increasing difficulty to manage and organize all this new information. The lack of a standard context representation scheme is one of the main struggles in this research area. Conventional methods for extracting knowledge from data rely on a standard representation or a priori relation, which may not be feasible for IoT and M2M scenarios. With this in mind we propose a stream characterization model in order to provide the foundations for a novel stream similarity metric. Complementing previous work on context organization, we aim to provide an automatic stream organizational model without enforcing specific representations. In this paper we extend our work on stream characterization and devise a novel similarity method.

Type of Paper and Keywords

Regular research paper: *stream mining, time series, machine learning, IoT, M2M, context awareness*

1 Introduction

Over the last years the Internet of Things (IoT) [34] has gained significant attention from both industry and academia. IoT has made it possible for everyday devices to acquire and store contextual data, and to use it at a later stage. This allows devices to share data with one another, in order to cooperate and accomplish a given objective. A cornerstone to this connectivity landscape is machine-to-machine (M2M) communications [10]. M2M generally refers to information and communication technologies able to measure, deliver, digest and react upon information autonomously, *i.e.* with none or minimal human interaction.

Context-awareness is an intrinsic property of IoT [27]. Context-aware communications and computing have played a critical role in understanding sensor data, since it provides the necessary tools to analyse data regarding an entity and choose a useful action. As discussed in [5] an entity's context can be used to provide added value: improve efficiency, optimize resources and detect anomalies, to name a few. However, recent projects follow a vertical approach [14, 28, 12], where devices/manufacturers can not share context information because each one uses its own structure, leading to information silos. This has hindered interoperability and the realisation of even more powerful IoT and M2M scenarios.

Context information is an enabler for further data analysis, potentially exploring the integration of an increasing number of untapped information sources. Not only are the common definitions of context information [1, 33] so broad that any data related to an entity can be considered context information, but they also do not provide any insight about the structure of context information. Currently there is no uniform way to share/understand vast amounts of IoT/M2M data, and it is unlikely that in the future a context representation standard will be widely adopted. First, there is the diversity of context representations, each of them designed for a specific usage and/or data types. Second, a widely adopted context representation does not completely solve the issue of knowledge extraction. Due to the vast amount of data being considered, it becomes extremely difficult to define *a priori* all the relations among information sources, patterns, and even possible optimizations.

Another important issue is the need for a new way to manage, store and process such diverse machine data: unconstrained, without limiting structures and with minimal human interaction. With this in mind we proposed a data organization model optimized for unstructured data [5, 4] that organizes context data based on semantic and stream similarity. Our model uses tailored features and unsupervised learning algorithms to automatically organize data. In this paper we extend our previous work on stream characterization model [18, 6] and devise a novel similarity metric for our stream model. Our generative model for stream characterization can be used either for stream generation or similarity.

The remainder of this paper is organized as follows. In Section 2 we discuss semantic similarity and present the most relevant methods. We discuss our generative model for stream characterization in Section 3. Details about the implementation of our prototype are given in Section 4. The results of our evaluation are in Section 5. Finally, the discussion and conclusions are presented in Section 6.

2 Related Work

Context information is an enabler for further data analysis, potentially exploring the integration of an increasing number of information sources. As previously mentioned, common definitions of context information [1, 33, 13] do not provide any insight about its structure. In fact, each device can share context information with a different structure. For example, sensory and location information can be used to characterize an entity's context, yet the two can have different structures. One important objective of context representation is to standardize the process of sharing and understanding context information. However, nowadays no widely accepted context representation scheme exists; instead there are several approaches to deal with context information. These can be divided into three groups: (i) adopt/create a new context representation, (ii) normalize the storing process through ontologies, (iii) accept the diversity of context representations.

In [25] the authors analyse two different projects related with context-awareness. One of the projects uses a single context representation scheme. The authors concluded that using a single context representation limits the relations that exist between all the data sources. As a consequence it becomes increasingly difficult to detect and react to complex events. Furthermore, it limits the quantity of data that can be shared with other projects.

The second possibility would be employing ontologies to normalize the organization process. Each context representation is mapped into the internal data model through an ontology [22]. This type of platform supports several context representations, yet it is necessary to define a new ontology (mapping) for each new representation. Defining a new ontology is a tedious task that requires human intervention. Due to the diversity and scale associated with IoT/ M2M scenarios it is extremely difficult to maintain this strategy. As an example, we can consider the lexical database WordNet [24]. WordNet is a manually-created hierarchical network of nodes (taxonomy), that due to funding and staffing issues is no longer accepting comments and suggestions. It is extremely difficult to maintaining large databases of relations (of any type) if they depend on human input.

As an alternative, we can accept the diversity of context representation as a consequence of economic pressures, and develop an efficient method to deal with it. Let us consider the specific case of IoT/M2M scenarios as a representative use case for context information and context-aware systems. In order to develop and deploy complex IoT/M2M scenarios we need to address the issues regarding storing, analysing and understanding IoT data. However, correctly managing IoT data has become a difficult task to accomplish. The volume and diversity of data puts a toll on conventional storage and analytical tools, restricting and limiting the development of complex IoT/M2M scenarios. Due to the volume and lack of formal representation, IoT data can be characterized as a combination of the unstructured data and Big Data paradigms. These paradigms are inherently connected, and are one of the factors that led to the advent of NoSQL databases [21, 9]. This insight points to the limitation of current technology when dealing with massive unstructured data.

Relational databases rely on predefined representations and *a priori* relations in order to correctly store and retrieve information. That is rather difficult to accomplish when the data is mostly unstructured, as is the case of IoT data. NoSQL databases relax some constraints and are good alternatives to several workloads and even small IoT scenarios. However, they lack advanced query capabilities, restricting the discovery of information and complex patterns [3, 5].

The limitations are not purely technological. Even if we were able to store and query all the data gathered by IoT devices, we would still need methods to organize, analyse and discover relevant relations between data sources and target functions. Most analytical tools rely on either *a priori* relations or a human to analyse the data. These elements bestow some latent knowledge to the underlying model, which implies top-down classification. Top-down classification limits the dimension along which one can make distinctions, and local choices at the leaves are constrained by global categorizations in the branches. It is therefore inherently difficult to put things in their hierarchical places, and the categories are often forced. Let us consider the following example. The information gathered from an accelerometer inside a vehicle can be used by city officials to detect potholes and other anomalies on the road. But it can also be used by policemen to detect dangerous manoeuvres and behaviours. These examples illustrate how difficult defining *a priori* relations in complex environments can become.

Some authors [29, 7, 15] point out that probabilistic models based on bottom-up characterization can produce better results than binary schemes based on top-down classification. Based on this approach we devised a bottom-up model [5] to organize context information without enforcing a specific representation. Our organization model is divided into two main parts, as depicted in Figure 1. The first part is composed by two components that represent the structured part of our model and account for the source identification and fixed d-dimensions respectively. These d-dimensions allow human users to select information based on time, location or even other dimensions, and can be understood as an OLAP cube helping in the process of filtering information. The second part represents machine learning features, that can be used to find similar or related sources of data. Up until now we have worked on semantic [5, 4] and stream features [18, 6]. In this paper we continue our work on stream similarity.

While there are several academic works based on stream prediction and mining [20], the same can not be said about stream similarity. Most methods are based on longest common sub-sequence algorithm [23, 8]. Some work related with detection patterns in time-series has been done in financial stock markets [17]. However, these methods are not ideal for generalized IoT/M2M data for two main reasons. First, data acquired from IoT devices tend to be noisy, can be shifted in time and have different scales. Second, the vast number of IoT devices implies that there are several streams for the same phenomenon. Our objective is to learn a representation of the phenomenon, combining all the streams in a single model. Due to these reasons we devised our own generative model. We are interested in characterizing the "shape" of a stream/time series, the closest analogue that we known being shape descriptors in image recognition, such as Roy's Shape Representation and Global Shape Context [26]. In fact our model draws inspirations from the previously mentioned techniques, since it also uses a grid like structure to capture the "shape" of a stream/time series.

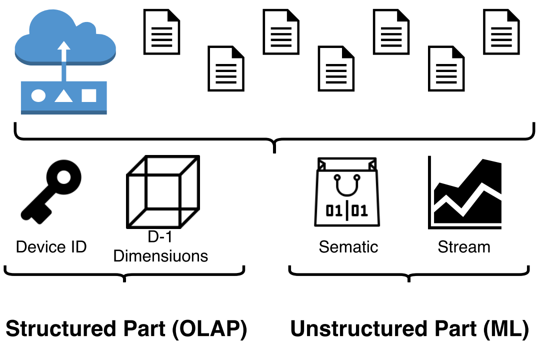


Figure 1: Context organization model based on semantic and stream similarity

It is not only IoT that will benefit from stream patterns' characterization. Any task that requires time-series clustering and/or classification will benefit from a stream characterization model. Typical real-world examples include financial data [2] and medical data [16]. Even areas, such as network optimization, BPMN execution flow and time-series privacy, may benefit from stream patterns' characterization. Using the hidden patterns in network traffic it is possible to extract more accurate network graphs and achieve better optimizations [30]. Data-flow errors in BPMN 2.0 process models, can be detected by mapping them to Petri Nets, unfolding the execution semantics and detect specific error patterns [32]. Time series anonymization is an important problem, by using a (n, l, k)-anonymity model that transform a time-series without jeopardizing the relevant patterns [19]. Another area that may benefit from stream characterization is model compression [11, 31]. By capturing the relevant characteristics of a time-series we can minimize the amount of information transmitted and stored. It can be specially useful for large IoT scenarios.

3 Generative Model for Stream Characterization

Before discussing the details of our stream characterization model, let us discuss its origin. With the advent of IoT/M2M devices, context-aware platforms require novel organizational models, learning algorithms and proper testing. However, it is rather difficult to evaluate the accuracy of these systems when the environment is as dynamic and vast as the IoT/M2M environment. In order to properly test these platforms we require both a controlled environment and tools to control the input data.

There are some possibilities, and the most common one is to use several datasets gathered from actual sensors. Gathering, pre-processing, classifying and maintaining these datasets requires human intervention and is time-consuming. Furthermore, in order to guarantee that the tests cover all (almost) the possible inputs, large amounts of data are required. One alternative to this is to develop a model that captures the information about a determined phenomenon and is able to generate its several instances that are statistically similar. This was the drive to develop our stream characterization model. Apart from stream generation, our model structure makes it ideal to develop similarity metrics. We intend to explore the full capabilities of this model, as a tailored feature for IoT/M2M organization, in future publications.

This section will address two different but related ideas. First, we will present our generative model for stream characterization based on Markov Chains and detail its inner workings. Second, we will elaborate on a stream generator that uses the previously mentioned model.

3.1 Stream Characterization

Our approach is to model a stream's behaviour using first order Markov chains. Considering a perfect scenario where there is no noise or errors, most events would thus happen in a very predictable manner (*i.e.* without major variances). We could formulate our model as Equation 1, by knowing how probable it is for, at a given time instant x_{i-1} with a value of y_j, a stream at the time x_i take a value of y_k. In other words, the probability of having value y_k at a time instant x_i knowing its immediate predecessor.

$$P_i(y_k|y_j) \tag{1}$$

For the remainder of this paper we will call the succession of a value to the one following it (along the x axis) a jump or transition.

We could then argue that using the method above and knowing all the probabilities of all the jumps along the period of the event, we could represent it with quite high confidence. For the sake of argument, consider that we had at our disposal such a probability function as expressed above, and we were given a sequence of values representing an event. We would like to compute the similarity (S) between the sequence of values and the probability function. This can be achieved by verifying all the values of P_i for all transitions within a sequence's period, and either averaging them or using some other statistical indicator to get a representative, normalized value of the overall resulting probabilities (see Equation 2, where n represents the number of samples in the stream).

$$S = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{2}$$

The probability function would assign an high or low value to each jump of the sequence based on how well it relates to the events expressed by the probability function itself. If the sequence's values were off the event's, then the overall probability would be low. On the other hand if it was high, then we could be confident that this sequence is similar to the event represented by the function.

The problem arises as we notice that this perfect scenario is not possible in practical cases, hence if we intend to use such a function as the one described above to represent a stream, we need to overcome three major issues and make a few changes to its definition:

1. Streams representing the same events may vary widely, for reasons such as noise, location, time of day, etc;
2. It is impractical, due to both time and space constraints, to have a function mapping every tuple of points $((x_i, y_j), (x_{i+1}, y_k))$ into a number (the probability of the transition);
3. Along the lines of the previous item, it is not reasonable to consider the continuous and/or infinite domain associated with most events (which would imply considering infinite values).

Our proposal attempts to solve these issues by overlaying a grid-like structure over the different values a stream takes along its period, effectively turning each (x_i, y_j) in the preceding discussion into a slot (as depicted in Figure 2). This gap gives rise to two other values that are now to be considered, Δx and Δy, each representing the resolution of their corresponding axis.

Issue 1 can be solved by overlaying multiple streams representing a same event, and computing the probabilities that arise from their transitions. Issues 2

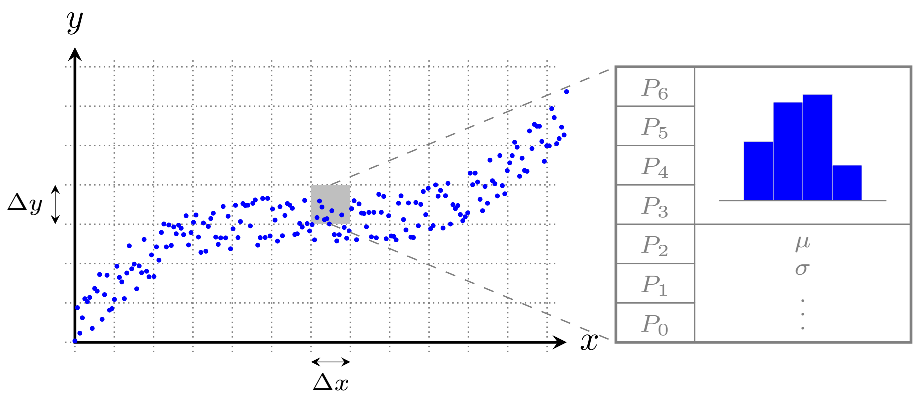


Figure 2: Structure proposed to model stream information
A grid is overlayed over the sample streams, in order to build a matrix like structure where each slot contains a probability vector, an histogram of values, and other relevant statistical values (*e.g.* the mean and standard deviation of the values inside the bin).

and partially 3 are solved by now considering jumps' areas instead of single values, in a sense discretizing both a stream's domain and codomain. By the law of large numbers and assuming that those streams do follow a pattern (even if with noise and/or erratic behaviour), one can be sure that eventually the probabilities will converge. Issue 3 can be further improved in the case of periodic streams.

Given that most real scenarios are periodic to some extent, the model can be constructed based on the event's period. In case the data has some seasonality property associated, a model can be contracted in accordance to each season's period. The period of the phenomenon is then taken as the domain of the grid described. This makes it possible to bear with the otherwise infinite domain of periodic streams. Each stream's period is taken as a 1-period stream by itself.

This way we are capable of characterizing the underlying behaviour of some event, based on the behavioural patterns of some related streams. We say this method is based on first order Markov chains since it assumes that there is little to no knowledge lost by only considering direct transitions along the x axis. This means that we do not use all the previous values a stream took before a given x_i when computing the probability of being in some other area in the time slot following (with $x_{i+1} \equiv x_i + \Delta x$). This is done to minimize the computational complexity that would arise from doing so.

The representation mentioned above can still have a problem: the notion of "area" itself. If it is too wide or too narrow, the model fails to capture the relevant pattern of the event. If any of Δx or Δy are too broad, information about the event will be lost. On the other hand, if these values are too narrow, the computation's complexity of the probabilities will start to degrade. Even worse, it can make the whole representation too specific (resulting in overfitting).

In order to minimize this issue we propose to keep the following values associated to each slot, as shown in Figure 2:

Probability vector: this is the function which makes possible representing the nature of the stream using probabilities. Each P_i maps to the probability of jumping to the y_i following along the x axis (the transition).

Histogram of values: each slot maintains a histogram of values, allowing the model to identify which values are more commonly found within that slot, minimizing the penalization of having large bins values. In a sense this adds another dimension to the model.

Other statistical values: other statistical values may be kept for further improvements. For example, keeping the average and the standard deviation of the values within the slot. These are both cheap computationally wise and may be of significance when evaluating how well a given point fits within the slot.

Our model also supports the generation of multiple continuum periods. We modelled this behaviour by computing the probabilities of wrapping around the matrix representation (*i.e.* going from the last column to the first). This way, with the same Markov simplification made throughout the document, we gave the model the knowledge to generate continuous stream (in a true infinite stream scenario). This property will become rather important when considering time shifts in a similarity metric.

3.2 Period Detection

In order to automate the usage of the generator (and later the similarity component), we have developed a module for automatic period detection. It works by first computing a periodogram of a stream, and selecting the k strongest frequencies from it — commonly named candidate frequencies. Then, for each of these candidates, an autocorrelation factor (ACF) with lag of the inverse of the frequency (the period) is computed. The period which gives the best ACF is selected.

Since the sample points of a stream may not be equally spaced, we drop portions of the stream that do not meet a minimum percentage of points given the period being considered. Furthermore, the portions which will be used are linearly interpolated in order to bring equal spacing between samples (which is important for the autocorrelation, otherwise the point-wise operations do not match). The interpolation is evaluated every $2\Delta t$, where Δt is the mean time difference along each original pair of succeeding sample points. Initially we did not use interpolation, making the autocorrelation fail due to the mismatch between samples' spacing. With linear interpolation the problem was solved.

3.3 Stream Generation

Apart from stream characterization and similarity estimation, our model can also be used to generate streams. Context-aware platforms, or indeed any platform that deals with context information (IoT/M2M data included), benefits from a realistic stream generator. As these platforms become smarter it also becomes imperative to validate and evaluate the platform in a controlled environment. In our specific case, initial work demanded the use of large datasets to carry on tests and to evaluate the capability of representation of our organization model. This lead to the development of a stream generator general enough to be used in a wide class of streams, which is used to build synthetic datasets from real ones we have, but which were not as big as needed.

Such generator would have to output plausible streams, and not just a stream which would for instance minimize the errors between itself and the set of streams given as examples. This constituted an opportunity to test our proposed representation. The internal structure of the generator is, thus, a matrix of slots, each with the values as described in Subsection 3.1. This matrix is built for each type of pattern we want to learn,

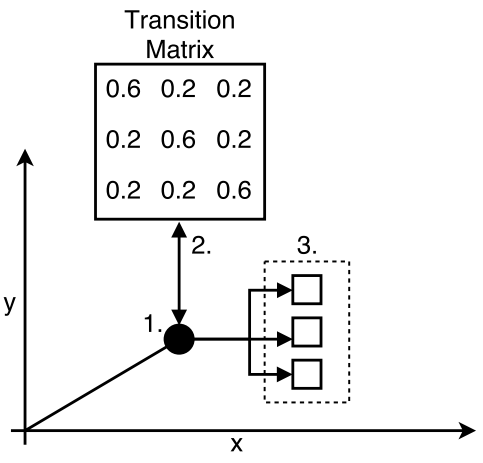


Figure 3: Generation process
1. At each gap, generate a random value that represents the transition probability; 2. Use the transition matrix to identify the next bin; 3. At the destination bin, generate a new value (based on its histogram).

from a set of streams representative of the pattern (*e.g.* temperature or humidity). After having the matrix built, we can traverse it (along its x axis) according to the probabilities and histograms found along the path in order to generate streams similar to the underlying pattern of the ones which were previously presented (as depicted in Figure 3).

Preliminary tests show how good is the capability of the generator to learn the most relevant motifs of the streams, being capable of generating realistic streams from the representation built. This is further discussed in Section 5.

3.4 Stream Similarity

For computing the similarity of a given stream when compared with a certain model, the stream is fitted into the model (similarly to what is done when building the model itself). Since at this stage we are dealing with a scale-dependant technique, streams and models whose domains are not directly comparable are not being considered.

The similarity value itself is obtained by traversing the stream and evaluating, at each transition, the likelihood (*i.e.* probability) of having both the stream's point at the bin being considered and the transition that the stream suggests.

Taking the definitions

P_{hi} := Normalized probability of point i of stream
P_{ti} := Normalized probability of the transition i

The similarity expression is then given by Equation 3, with both sums running over all the stream's transitions (and hence points).

$$S = \frac{\sum P_{ti} \cdot P_{hi}}{\sum P_{ti}}, \tag{3}$$

In case there is no bin (in the model) to characterize the point being considered at a given time, then $P_h = 0$ and $P_t = 1$ for that parcel. Otherwise, if the bin exists but there is no possible transition, then $P_h = 0$ and P_t is taken as the (normalized) probability of the strongest transition of the bin. This measures are taken so as to penalize to different degrees the stream's parcels which cannot be compared.

4 IMPLEMENTATION

So far some ideas presented in this paper have appeared, in a sense, as isolated units. With this in mind, the goal of this section is to describe how they are brought together, answering two major scenarios: stream generation and stream similarity. Usual stages of machine learning pipelines (*e.g.* preprocessing stages where outliers are removed) are omitted, so as to keep the text concise and centered around the ideas previously presented.

The first algorithm that shall be presented, Algorithm 1, builds a model from a set of streams. The function `FindResolutionOf` is still only theoretical. The resolutions that were used during testing were found by experimentation. Also, the goal of the function `SnapToResoltution` is to fit each stream into the grid that is being built, for example deciding in which bin each portion of the stream fits. This is later used when computing the probabilities of each transition.

The next algorithm to be presented is Algorithm 2, which given a model generates a stream. The function `GeneratePoint` uses the histogram of a model's bin to generate a point in accordance with its distribution. The function `GenerateNextBin` uses the probability vector of a bin to determine where the generated stream will flow through. Both of these notions are discussed in Subsection 3.3.

Finally, the similarity algorithm, Algorithm 3, is described, which assigns a similarity score for a stream against a model. Even though it is stated that the initialization is similar to the first lines of Algorithm 1, both the period and resolution values used are the ones of the model. Despite this, they are still computed for the stream itself. They are compared against the respective values associated with the model, and in case they are not comparable within certain bounds, the similarity value is penalized. Also, exception handling is not included in this description for brevity. For example, in case a stream's value is not present in the model's bin that it is being compared with, then default values are used instead (which are mentioned in Subsection 3.4).

As a final note, the actual implementation of these methods and associated data structures was carried in `Python3`, resorting mainly to the standard libraries, `numpy` and `scipy`.

Algorithm 1 Model Building

```
1: function BUILDMODEL($streams$)
2:     $period \leftarrow FindPeriod(streams)$
3:     $SplitStreamsByPeriod(streams, period)$
4:     $\Delta x,\ \Delta y \leftarrow FindResolutionOf(streams)$
5:     for all $stream \in streams$ do
6:         $SnapToResoltution(stream, \Delta x, \Delta y)$
7:     end for
8:     $model \leftarrow ComputeProbabilities(streams)$
9:     return $model$
10: end function
```

Algorithm 2 Stream Generation

```
1: function GENERATESTREAM($model,\ yinit$)
2:     $bin \leftarrow (0,\ yinit)$
3:     $genstream \leftarrow \{GeneratePoint(model, bin)\}$
4:     for $i \leftarrow 1, \#ColumnsOf(model) - 1$ do
5:         $bin \leftarrow GenerateNextBin(model, bin)$
6:         $genstream \leftarrow$
                   $genstream$
                   $+$
                   $\{GeneratePoint(model, bin)\}$
7:     end for
8:     return $genstream$
9: end function
```

Algorithm 3 Stream Similarity

```
1: function SIMILARITY($model,\ stream$)
       % Initialization as in lines 2 to 7 of Algorithm 1
2:     $m1sum \leftarrow 0$
3:     $m2sum \leftarrow 0$
4:     for $i \leftarrow 0, \#ColumnsOf(model) - 1$ do
5:         $t \leftarrow P_t(model, stream, i)$
6:         $h \leftarrow P_h(model, stream, i)$
7:         $m1sum \leftarrow m1sum + t \cdot h$
8:         $m2sum \leftarrow m2sum + t$
9:     end for
10:    return $m1sum/m2sum$
11: end function
```

5 Performance evaluation

This section will present our current results. First, we evaluate the period detection algorithm. Second, we present the evaluation of stream generation based upon our model. Finally, we evaluate the accuracy of our stream similarity metric. We used a home automation dataset[1] to evaluate our model, it was composed of three different kinds of natural phenomena: environment temperature, humidity and light intensity. Each set considered was composed of approximately one hundred streams.

5.1 Period Detection

We expected a periodicity of approximately 1 day (around 86400 s) for each phenomenon, which is further suggested by visual inspection of the plots of the streams we used. Figure 4 shows three histograms, each depicting the computed periods over approximately fifty streams of the respective phenomena. Although including some outliers, we highlight that the maximums of each plot stand well above the rest of the values and are indeed close (within a 5% margin) to the expected 1 day period.

5.2 Stream Generation

We use MSE (mean-square error) and visual representations to evaluate the generative performance of our model. This evaluation was carried by k-cross validation, and is displayed at Table 1. We obtained these values by selecting one real stream and comparing it with all the others, and doing the same for a generated stream (and repeating selecting/generating other streams). It is interesting to see that the differences between the real and generated results are not far off. Meanwhile, Figure 5 enables a more visual evaluation of our results, plotting real *vs* generated streams.

We would like to highlight that not only are the curves similar, but the standard deviation at each point is also comparable. This suggests that our model does not seem to be over fitting — the set of learning curves was composed of heterogeneous samples, which is indeed propagated to the generated streams. The MSE values also validate that our generated curves are not too far off the real ones. Even regarding "Light", which scored a much bigger MSE than the other sets, our model agrees with the results from real streams.

Our model also supports the generation of multiple periods. We have called this the "continuum" mode, which Figure 6 presents a plot of. We find it relevant to say that the transitions between periods are smooth and that, without the colouring to tell them apart, the transition points would probably be unnoticeable.

[1] available at `http://db.csail.mit.edu/labdata/labdata.html`

Table 1: MSE values computed for the streams generated

	Real		
	Mean	Median	Stdev
Temperature	10.5	9.3	3.9
Humidity	51.4	37.3	27.9
Light	217360	175633	100361
	Generated		
	Mean	Median	Stdev
Temperature	10.0	9.3	3.0
Humidity	48.3	49.1	11.8
Light	221271	222265	39933

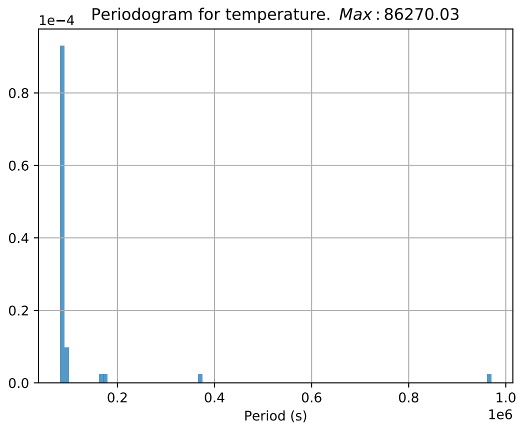
Table 2: Similarity scores obtained

Stream	Model	Similarity
humidity	humidity	$\mathbf{0.68 \pm 0.03}$
	light	0.12 ± 0.04
	temperature	0.03 ± 0.04
light	humidity	0.00 ± 0.00
	light	$\mathbf{0.71 \pm 0.05}$
	temperature	0.00 ± 0.00
temperature	humidity	0.11 ± 0.05
	light	0.08 ± 0.01
	temperature	$\mathbf{0.67 \pm 0.03}$

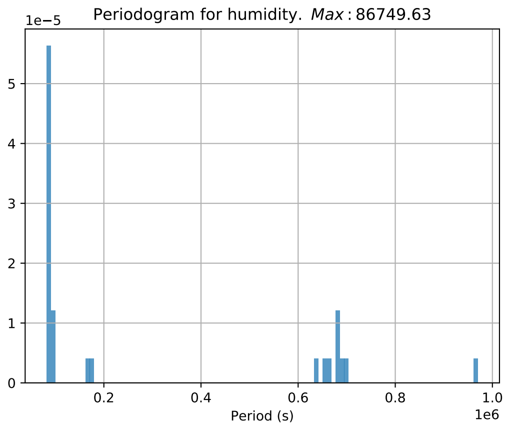
5.3 Stream Similarity

The stream similarity technique previously discussed (Subsection 3.4) was tested by continuously selecting two different features. A stream of each feature would be picked, with the remaining streams being used to build a model. Finally, each of the streams would be matched against each of the models (including streams and models relative to the same feature), with the similarity of the stream to the model being computed. This procedure was repeated until around two thousand stream-model matches were obtained. The results are illustrated in Table 2.

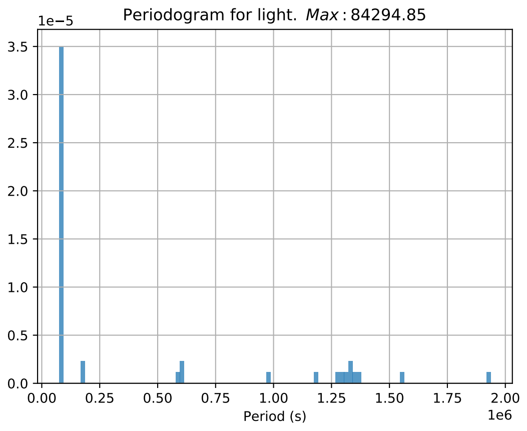
As it can be seen, there is an undeniable abyss between correct stream-model matches and incorrect ones. Following this test a threshold could easily be built, which would predict with very high confidence whether a stream was or not a match to a model. In the future we intend to test with more features in order to verify



(a) Temperature



(b) Humidity



(c) Light

Figure 4: Periodograms for the three phenomena analysed

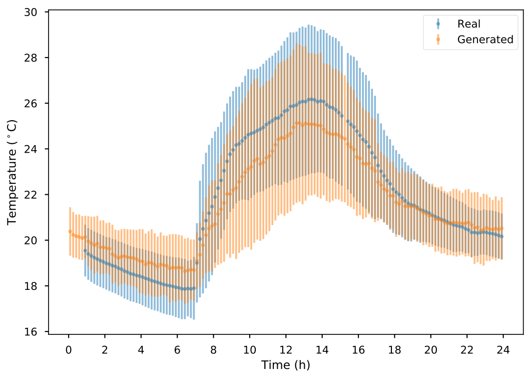
whether this pattern holds.

Despite these results it is noticeable that none of the correct matches produced a score above 0.80, which was our expectation when we first thought of this test. This can be justified by the disparity that streams naturally hold, and which were brought to light in past experiments (Table 1, "real" row). With this in mind, one cannot expect a stream to match a model with extremely high similarity, since the model itself is built to bare with the (many) natural differences of the streams. Extremely high values of similarity would also indicate that the model was overfitting. Further validations need to be carried in order to better verify this, but nonetheless for the time being this is an idea that seems to be sound.

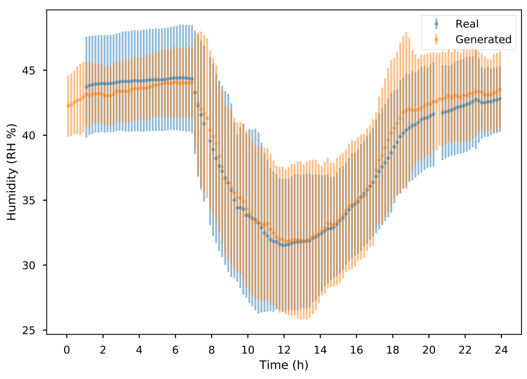
6 Summary and Conclusions

The number of sensing devices is increasing at a steady step. Each one of them generates massive amounts of information. However, each device/manufacture shares context information with different structures, hindering interoperability in IoT/M2M scenarios.

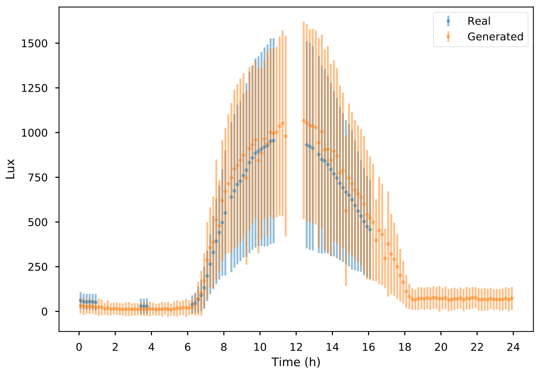
We tackled this issue by developing an organization model agnostic to context representation. Our organization model uses tailored features to automatically organize data and improve its accuracy. By using our generative stream model as a tailored feature to describe stream patterns we believe that our organization model will be further improved. It is worthwhile to mention that there are several academic works based on stream prediction and mining [20], but the same cannot be said about stream similarity and stream characterization. Further work needs to be done



(a) Temperature



(b) Humidity



(c) Light

Figure 5: The three kinds of generated streams: temperature, humidity and light
The vertical bars represent the standard deviation (at each point) of 20 different streams.

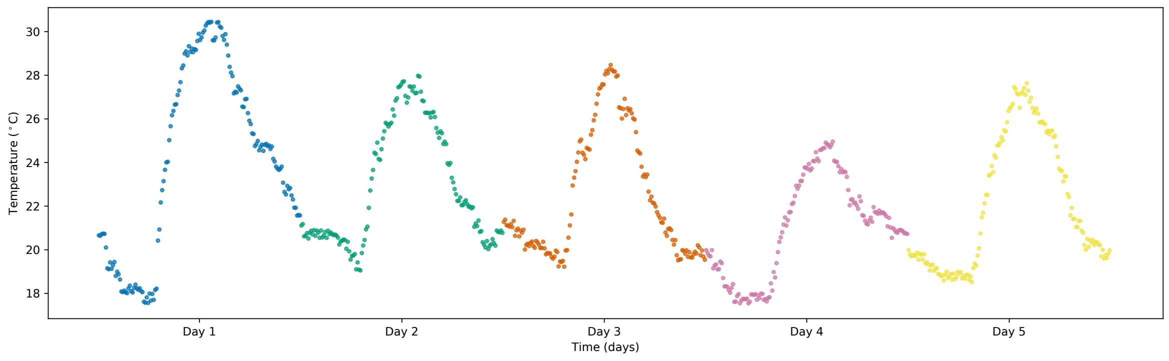


Figure 6: 5 temperature streams generated as a continuum
The generator had been trained with around 100 streams prior to the generation.

to assert some ideas expressed in this paper, but our stream characterization model appears to be a viable option.

We are currently devising a similarity metric to estimate the similarity between two stream models. Being able to analyse if two different stream models are similar allow us to better organize context information based on similarity. This allows us to organize data based not only on semantic features [4], but also on stream patterns. Furthermore, our model will serve as a strong filter, trimming the search space so that more advanced techniques can used. For example, IoT/M2M platforms can use machine learning techniques over our context organization model to provide smart and proactive services, high level inference, amongst others.

There is room to further improve our stream characterization model. Specially to cope with the variability associated with IoT/M2M scenarios. Some questions which are yet to be answered include: Is scale (along the y axis) important? If yes, in which cases and how to work with it? How to cope with time and location differences across the different sensors? We will continue our research on these topics and hopefully answer these questions in future publications.

Meanwhile, the ability to generate streams resembling a given set of learning ones can be useful in many situations. For instance, to generate large synthetic datasets where otherwise there is no specific generator available. Our general purpose generator has another big advantage, since it improves the repeatability and validity of IoT/M2M and context-aware platforms. Currently these platforms use advanced machine learning algorithms to improve and optimize several processes. Having the ability to test them for a long time in a controlled environment is extremely important.

Acknowledgements

The present study was developed in the scope of the **Smart Green Homes** Project [POCI-01-0247-FEDER-007678], a co-promotion between **Bosch Termotecnologia S.A.** and the **University of Aveiro**. It is financed by Portugal 2020 under the Competitiveness and Internationalization Operational Program, and by the European Regional Development Fund. This work was also partially supported by research grant SFRH/BD/94270/2013.

References

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proc. of the 1st international symposium on Handheld and Ubiquitous Computing*, 1999, pp. 304–307.

[2] S. Aghabozorgi and Y. W. Teh, "Stock market co-movement assessment using a three-phase clustering method," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1301–1314, 2014.

[3] M. Antunes, D. Gomes, and R. Aguiar, "Context storage for m2m scenarios," in *Communications (ICC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3664–3669.

[4] M. Antunes, D. Gomes, and R. Aguiar, "Learning semantic features from web services," in *Future Internet of Things and Cloud (FiCloud), 2016 4rd International Conference on*. IEEE, 2016.

[5] M. Antunes, D. Gomes, and R. L. Aguiar, "Scalable semantic aware context storage," *Future Generation Computer Systems*, vol. 56, pp. 675–683, Mar. 2016.

[6] M. Antunes, R. Jesus, D. Gomes, and R. Aguiar, "Improve iot/m2m data organization based on stream patterns," in *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 2017.

[7] G. Avram, "At the crossroads of knowledge management and social software," *Electronic Journal of Knowledge Management*, vol. 4, no. 1, pp. 1–10, January 2006.

[8] A. Camerra, J. Shieh, T. Palpanas, T. Rakthanmanon, and E. Keogh, "Beyond one billion time series: indexing and mining very large time series collections with iSAX2+," *Knowledge and Information Systems*, vol. 39, no. 1, pp. 123–151, 2014.

[9] R. Cattell, "Scalable sql and nosql data stores," *SIGMOD Rec.*, vol. 39, no. 4, pp. 12–27, May 2011.

[10] K.-C. Chen and S.-Y. Lien, "Machine-to-machine communications: Technologies and challenges," *Ad Hoc Networks*, vol. 18, pp. 3–23, 2014.

[11] M. Danieletto, N. Bui, and M. Zorzi, "Improving internet of things communications through compression and classification," in *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, March 2012, pp. 284–289.

[12] S. K. Datta, C. Bonnet, R. P. F. D. Costa, and J. Härri, "Datatweet: An architecture enabling

data-centric iot services," in *2016 IEEE Region 10 Symposium (TENSYMP)*, May 2016, pp. 343–348.

[13] A. K. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, 2001.

[14] R. Fantacci, T. Pecorella, R. Viti, and C. Carlini, "Short paper: Overcoming iot fragmentation through standard gateway architecture," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, March 2014, pp. 181–182.

[15] T. Gruber, "Ontology of folksonomy: A mash-up of apples and oranges," *International Journal on Semantic Web and Information Systems*, vol. 3, no. 2, pp. 1–11, 2007.

[16] S. Hirano and S. Tsumoto, "Cluster analysis of time-series medical data based on the trajectory representation and multiscale comparison techniques," in *Sixth International Conference on Data Mining (ICDM'06)*, Dec 2006, pp. 896–901.

[17] S. Jeon, B. Hong, and V. Chang, "Pattern graph tracking-based stock price prediction using big data," *Future Generation Computer Systems*, 2017.

[18] R. Jesus, M. Antunes, D. Gomes, and R. Aguiar, "Extracting knowledge from stream behavioural patterns," in *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security*. SCITEPRESS - Science and Technology Publications, 2017.

[19] S. Kessler, E. Buchmann, T. Burghardt, and K. Böhm, "Pattern-sensitive time-series anonymization and its application to energy-consumption data," *Open Journal of Information Systems (OJIS)*, vol. 1, no. 1, pp. 3–22, 2014. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:101:1-201705194696

[20] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, and J. Stefanowski, "Open challenges for data stream mining research," *SIGKDD Explor. Newsl.*, vol. 16, no. 1, pp. 1–10, Sep. 2014.

[21] N. Leavitt, "Will nosql databases live up to their promise?" *Computer*, vol. 43, no. 2, pp. 12–14, February 2010.

[22] P. Lopes and J. L. Oliveira, "Coeus: Semantic web in a box for biomedical applications," *Journal of Biomedical Semantics*, vol. 3, no. 1, p. 11, 2012.

[23] A. Marascu, S. A. Khan, and T. Palpanas, "Scalable similarity matching in streaming time series," in *Advances in Knowledge Discovery and Data Mining: 16th Pacific-Asia Conference, PAKDD 2012 Proceedings, Part II*. Springer Berlin Heidelberg, June 2012, pp. 218–230.

[24] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, November 1995.

[25] T. Mota, N. Baker, B. Moltchanov, R. Ioanna, and K. Frank, "Towards pervasive smart spaces: A tale of two projects," in *Future Network & Mobile Summit 2010. The Second International Workshop on Information Quality and Quality of Service for Pervasive Computing in Conjunction with IEEE PERCOM 2010*, 2010.

[26] R. Pereira and L. Seabra Lopes, *Learning Visual Object Categories with Global Descriptors and Local Features*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 225–236.

[27] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.

[28] J. Robert, S. Kubler, Y. L. Traon, and K. Främling, "O-mi/o-df standards as interoperability enablers for industrial internet: A performance analysis," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, October 2016, pp. 4908–4915.

[29] C. Shirky, "Ontology is overrated: Categories, links, and tags," http://shirky.com/writings/ontology_overrated.html, May 2005, accessed: 22-07-2013.

[30] G. Sun, V. Chang, G. Yang, and D. Liao, "The cost-efficient deployment of replica servers in virtual content distribution networks for data fusion," *Information Sciences*, 2017.

[31] A. Ukil, S. Bandyopadhyay, and A. Pal, "Iot data compression: Sensor-agnostic approach," in *2015 Data Compression Conference*, April 2015, pp. 303–312.

[32] S. von Stackelberg, S. Putze, J. Mülle, and K. Böhm, "Detecting data-flow errors in bpmn 2.0," *Open Journal of Information Systems (OJIS)*, vol. 1, no. 2, pp. 1–19, 2014. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:101:1-2017052611934

[33] T. Winograd, "Architectures for context," *Hum.-Comput. Interact.*, vol. 16, no. 2, pp. 401–419, December 2001.

[34] F. Wortmann, K. Flüchter *et al.*, “Internet of things,” *Business & Information Systems Engineering*, vol. 57, no. 3, pp. 221–224, 2015.

AUTHOR BIOGRAPHIES

Ricardo Jesus is a MSc student in Computers and Telematics Engineering from the University of Aveiro. He received his bachelor’s in Computers and Telematics Science from the same university in 2017. His current research interests include subjects such as number theory and artificial intelligence (namely machine learning). He has been a member of ATNoG, a research group of the Instituto de Telecomunicações (Telecommunications Institute) since 2015. At ATNoG his main research has been targeted towards pattern characterization using probabilistic models, being involved in projects such as TVPulse and SCoT.

Mário Antunes received his computer and telematics engineering M.Sc. degree in 2011, with first class honors, from the Electronics, Telecommunication and Informatics Department, University of Aveiro, Portugal. He is currently working as a researcher for the Advanced Telecommunications and Networks Group at IT-Aveiro. His main research areas focus on Knowledge Extraction and Context Storage in Internet of Things (IoT) Scenarios using Machine Learning techniques and Big Data repositories. His works include developing efficient ways to deal with unstructured information. These techniques are being implemented and evaluated in advance machine-to-machine projects, such as APOLLO and SCOT.

Diogo Gomes graduated in Computers and Telematics Engineering from the University of Aveiro in 2003 with first class honors, and concluded his Ph.D. by the same University on Resource Optimization for Broadcast Networks in 2009. He is currently an Auxiliar Professor at the University of Aveiro. In the last 15 years has participated in several EU funded projects such as IST-Mobydick, IST-Daidalos, IST-Akogrimo, IST-C-MOBILE, ICT-C-Cast, ICT-Onelab2 and ICT-Medieval where besides conducting research on QoS, IP Mobility, Multicast/Broadcast and Service & Application Development has always been deeply involved in the deployment of prototypes and demonstrations. Recently his research interest are related to Knowledge Extraction and Context Storage in Internet of Things (IoT) Scenarios using Machine Learning techniques and Big Data repositories.

Rui L. Aguiar is full Professor at the University of Aveiro where he received his Ph.D. degree in 2001 in electrical engineering. He has been an adjunct professor at the INI, Carnegie Mellon University and is invited researcher at Universidade Federal de Uberlandia. His current research interests are centered on advanced communication systems and he has more than 400 published papers. He is a member of the steering Board of the Networld 2020 ETP. He has served as Technical and General Chair of multiple conferences and is Associate Editor of several journals. He is a member of ACM and a senior member of IEEE.