SO_MAD: SensOr Mining for Anomaly Detection in Railway Data

Julien Rabatel, Sandra Bringay, Pascal Poncelet

SO_MAD: SensOr Mining for Anomaly Detection in railway data

Julien Rabatel[1,2], Sandra Bringay[1,3], and Pascal Poncelet[1]

[1] LIRMM, Université Montpellier 2, CNRS
161 rue Ada, 34392 Montpellier Cedex 5, France
[2] Fatronik France Tecnalia, Cap Omega, Rond-point Benjamin Franklin - CS 39521
34960 Montpellier, France
[3] Dpt MIAp, Université Montpellier 3, Route de Mende
34199 Montpellier Cedex 5, France {rabatel,bringay,poncelet}@lirmm.fr

Abstract. Today, many industrial companies must face problems raised by maintenance. In particular, the anomaly detection problem is probably one of the most challenging. In this paper we focus on the railway maintenance task and propose to automatically detect anomalies in order to predict in advance potential failures. We first address the problem of characterizing normal behavior. In order to extract interesting patterns, we have developed a method to take into account the contextual criteria associated to railway data (itinerary, weather conditions, etc.). We then measure the compliance of new data, according to extracted knowledge, and provide information about the seriousness and possible causes of a detected anomaly.

1 Introduction

Today, many industrial companies must face problems raised by maintenance. Among them, the anomaly detection problem is probably one of the most challenging. In this paper we focus on the railway maintenance problem and propose to automatically detect anomalies in order to predict in advance potential failures. Usually data is available through sensors and provides us with important information such as temperatures, accelerations, velocity, etc. Nevertheless, data collected by sensors are difficult to exploit for several reasons. First because, a very large amount of data usually available at a rapid rate must be managed. Second, they contain a very large amount of data to provide a relevant description of the observed behaviors. Furthermore, they contain many errors: sensor data are very noisy and sensors themselves can become defective. Finally, when considering data transmission, very often lots of information are missing.

Recently, the problem of extracting knowledge from sensor data have been addressed by the data mining community. Different approaches focusing either on the data representation (e.g., sensors clustering [1], discretization [2]) or knowledge extraction (e.g., association rules [2], [3], [4], [5], sequential patterns [6], [7], [8]) were proposed. Nevertheless, they usually do not consider that contextual

information could improve the quality of the extracted knowledge. The development of new algorithms and softwares is required to go beyond the limitations. We propose a new method involving data mining techniques to help the detection of breakdowns in the context of railway maintenance. First, we extract from sensor data useful information about the behaviors of trains and then we characterize normal behaviors. Second, we use the previous characterization to determine if a new behavior of a train is normal or not. We are thus able to automatically trigger some alarms when predicting that a problem may occur.

Normal behavior strongly depends on the context. For example, a very low ambient temperature will affect a train behavior. Similarly, each itinerary with its own characteristics (slopes, turns, etc..) influences a journey. Consequently it is essential, in order to characterize the behavior of trains as well as to detect anomalies, to consider the surrounding context. We have combined these elements with data mining techniques. Moreover, our goal is not only to design a system for detecting anomalies in train behavior, but also to provide information on the seriousness and possible causes of a deviation.

This paper is organized as follows. Section 2 describes the data representation in the context of train maintenance. Section 3 shows the characterization of normal behaviors by discovering sequential patterns. Experiments conducted with a real dataset are described in Section 5. Finally, we conclude in Section 6.

2 Data Representation

In this section, we address the problem of representing data. From raw data collected by sensors, we design a representation suitable for data mining tasks.

2.1 Sensor Data for Train Maintenance

The data resulting from sensors for train maintenance is complex for the two following reasons: *(i)* very often errors and noisy values pervades the experimental data; *(ii)* multisource information must be handled at the same time. For instance, in train maintenance following data must be considered.

Sensors. Each sensor describes one property of the global behavior of a train which can correspond to different information (e.g., temperature, velocity, acceleration).

Measurements. They stand for numerical values recorded by the sensors and could be very noisy for different reasons such as failures, data transfer, etc.

Readings. They are defined as the set of values measured by all the sensors at a given date. The information carried out by a reading could be considered as the state of the global behavior observed at the given moment. Due to the data transfer some errors may occur and then readings can become incomplete or even missing.

We consider that the handled data are such as those described in Table 1, where a **reading** for a given date (first column) is described by **sensor measures** (cells of other columns).

TIME	Sensor 1	Sensor 2	Sensor 3	...
2008/03/27 06: 36: 39	0	16	16	...
2008/03/27 06: 41: 39	82.5	16	16	...
2008/03/27 06: 46: 38	135.6	19	21	...
2008/03/27 06: 51: 38	105	22	25	...

Table 1. Extract from raw data resulting from sensors.

2.2 Granularity in Railway Data

Data collected from a train constitutes a list of readings describing its behavior over time. As such a representation is not appropriate to extract useful knowledge, we decompose the list of readings at different levels of granularity and then we consider the three following concepts journeys, episodes and episode fragments which are defined as follows.

Journey. The definition of a journey is linked to the railway context. For a train, a journey stands for the list of readings collected during the time interval between the departure and the arrival. Usually, a journey is several hours long and has some interruptions when the train stops in railway stations. We consider the decomposition into journeys as the coarsest granularity of railway data.

Let $minDuration$ a minimum duration threshold, $maxStop$ a maximum stop duration, and J be a list of readings $(r_m, ..., r_i, ...r_n)$, where r_i is the reading collected at time i. J is a journey if:

1. $(n - m) > minDuration$,
2. $\nexists (r_u, ..., r_v, ...r_w) \subseteq J \mid \begin{cases} (w - u) > maxStop, \\ and\ \forall\ v \in [u, w],\ velocity(v) = 0. \end{cases}$

Episode. The main issue for characterizing train behavior is to compare elements which are similar. However, as trains can have different routes the notion of journey is not sufficient (for instance, between two different journeys, we could have different number of stops as well as a different delay between two railway stations). That is the reason why we segment the journeys into episodes to get a finer level of granularity. To obtain the episodes, we rely on the stops of a train (easily recognizable considering the train velocity).

An episode is defined as a list of readings $(r_m, ...r_i, ..., r_n)$ such as:

- $velocity(m) = 0$ and $velocity(n) = 0$ [1],
- if $m < i < n$, $velocity(i) \neq 0$.

[1] Here, the velocity of the train at time t is denoted as $velocity(t)$.

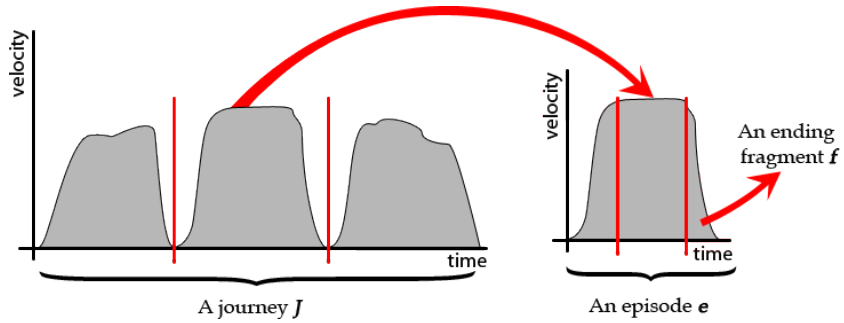

Fig. 1. Segmentation of a journey into episodes.

Figure 1 describes a segmentation of a journey into episodes by considering the velocity changes. This level of granularity is considered as the most relevant because it provides us with a set of homogeneous data. However, we can segment episodes in order to obtain a more detailed representation and a finer granularity level.

Episode Fragment. The level of granularity corresponding to the fragments is based on the fact that the behavior of a train during an episode can easily be divided in three chronological steps. First, the train is stationary (i.e., *velocity* 0) then an acceleration begins. We call this step the *starting* step. More formally, let $E = (r_m, ..., r_n)$ be an episode. The $starting fragment$ $E_{starting} = (r_m, ...r_k)$ of this episode is a list of readings such as:

$$\forall i, j \in [m, k],\ i < j \Leftrightarrow velocity(i) < velocity(j).$$

At the end of an episode, the train begins a deceleration ending with a stop. This is the *ending step*. More formally, let $E = (r_m, ..., r_n)$ be an episode. The $ending fragment$ $E_{ending} = (r_k, ...r_n)$ of this episode is a list of readings such as:

$$\forall i, j \in [k, n],\ i < j \Leftrightarrow velocity(i) > velocity(j).$$

The *traveling fragment* is defined as the sublist of a given episode between the *starting fragment* and the *ending fragment*. During this fragment, there are accelerations or decelerations, but no stop. More formally, let E be an episode, $E_{starting}$ its starting fragment, and E_{ending} its ending fragment. Then, the *traveling fragment* of E, denoted as $E_{traveling}$, is a list of readings defined as :

$$E_{traveling} = E - E_{starting} - E_{ending}.$$

Figure 1 shows the segmentation of an episode into three fragments: the starting fragment, the traveling fragment and the ending fragment.

From now we thus consider that all the sensor data are stored in a database, containing all information about the different granularity levels. For example, all the sensor readings composing the fragment shown in Figure 1 are indexed and we know that a particular fragment f is an ending fragment included in an episode e, belonging to the journey J. J is associated with the itinerary I and the index of e in I is 2 (i.e., the second portion of this route).

3 Normal Behavior Characterization

In this section, we focus on the data mining step in the knowledge discovery process and more precisely on the extraction of patterns characterizing normal behaviors.

3.1 How to Extract Normal Behavior?

The objective of the behavior characterization is, from a database of sensor measurements, to provide a list of patterns depicting normal behavior. We want to answer the following question: *which patterns often appear in the data?* Such a problem, also known as pattern mining, has been extensively addressed by the data mining community in the last decade.

Among all data mining methods, we can cite the sequential patterns mining problem. The sequential patterns were introduced in [9] and can be considered as an extension of the concept of association rule [10] by handling timestamps associated to items. The research for sequential patterns is to extract sets of items commonly associated over time. In the " *basket market* " concern, a sequential pattern can be for example: " *40 % of the customers buy a television, then buy later on a DVD player* ". In the following we give an overview of the sequential pattern mining problem.

Given a set of distinct attributes, an *item*, denoted as i, is an attribute. An itemset, denoted as I, is an unordered collection of items ($i_1 i_2 ... i_m$). A sequence, denoted as s, is an ordered list of itemsets $\langle I_1 I_2 ... I_k \rangle$. A sequence database, denoted as DB, is generally a large set of sequences. Given two sequences $s = \langle I_1 I_2 ... I_m \rangle$ and $s' = \langle I'_1 I'_2 ... I'_n \rangle$, if there exist integers $1 \leq i1 < i2 < ... < im \leq n$ such that $I_1 \subseteq I'_{i_1}$, $I_2 \subseteq I'_{i_2}$, ..., $I_m \subseteq I'_{i_m}$, then the sequence s is a *subsequence* of the sequence s', denoted as $s \sqsubseteq s'$.

The *support* of a sequence is defined as the fraction of total sequences in DB that support this sequence. If a sequence s is not a subsequence of any other sequences, then we say that s is *maximal*.

A sequence is said to be frequent if its support is at greater than or equal to a threshold minimum support ($minSupp$) specified by the user.

The sequential patterns mining problem is, for a given threshold $minSupp$ and a sequence database DB, to find all maximum frequent sequences.

Sequential Patterns and Sensor Data. The discovery of sequential patterns in sensor data in the context of train maintenance requires choosing a data format adapted to the concepts of sequence, itemsets and items defined earlier.

So from now we consider a sequence as a list of readings, an itemset as a reading, and an item as the state of a sensor. The order of itemsets in a sequence is given by the timestamps associated to each reading.

Items are Si_{vt}, where Si is a sensor and vt is the value measured by the sensor at time t. For example, data described in Table 1 are translated into the following sequence:

$$\langle (S1_0 S2_{16} S3_{16})(S1_{82.5} S2_{16} S3_{16})(S1_{135.6} S2_{19} S3_{21})(S1_{105} S2_{22} S3_{25})\rangle.$$

In addition, we use generalized sequences and time constraints ([11], [12]). More precisely, a time constraint called *maxGap* is set in order to limit the time between two consecutive itemsets in a frequent sequence. For instance, if *maxGap* is set to 15 minutes, the sequence $\langle (S1_{low})(S2_{low}, S3_{high})\rangle$ means that the state described by the second itemset occurs at most 15 minutes after the first one.

A sequence corresponds to a list of sensor readings. So, a sequence database can be created, where a sequence is a journey, an episode or an episode fragment, depending on the chosen level of granularity (see Section 2).

3.2 Contextualized Characterization

	Environmental Dimensions		Structural Dimensions	
id	Duration	Exterior Temperature	Route	Index
e_1	low	high	J1	E1
e_2	low	low	J1	E2
e_3	high	high	J2	E1
e_4	low	low	J1	E1
e_5	high	low	J1	E2

Table 2. Episodes and contextual information.

Structural and Environmental Criteria

With the data mining techniques described in the previous section we are able to extract patterns describing a set of episodes which currently occurs together. However, they are not sufficient to accurately characterize train behaviors. Indeed, the behavior of a train during a trip depends on contextual[2] criteria. Among these criteria, we can distinguish the two following categories:

[2] The notion of context stands for the information describing the circumstances in which a train is traveling. This information is different from behavioral data that describe the state of a train.

- **Structural criteria**, providing information on the journey structure of the studied episode (route[3], episode index in the route).
- **Environmental criteria**, providing information on the contextual environment (weather conditions, travel characteristics, etc.).

Example 1 *Table 2 presents a set of episodes, identified by the id column. In this example, the sensor data were segmented by selecting the level of granularity corresponding to episodes (see Section 2). Each episode is associated with environmental criteria (the duration of the episode and the exterior temperature) and structural criteria (the global route, and the index of the episode in this route). For example, the duration of the episode* e_2 *is short, and this episode was done with a low exterior temperature. In addition,* e_2 *is part of the itinerary denoted by* $J1$*, and is the second portion of* $J1$*.*

Let us consider a more formal description of the context of an episode. Each episode is described in a set of n dimensions, denoted by D. There are two subsets D_E and D_S of D, such as:

- D_E is the set of environmental dimensions. In Table 2, there are two environmental dimensions: *Duration* and *Exterior Temperature*.
- D_S is the set of structural dimensions, i.e., the *Route* dimension whose value is the route of the episode, and the dimension *Index* for the index episode in the overall route.

Data Mining and classes

Now we present how the influence of these criteria on railway behaviors are handled in order to extract knowledge. The general principle is the following: *(i)* we divide the data into classes according to criteria listed above and *(ii)* we extract frequent sequences in these classes in order to get contextualized patterns.

Based on the general principle of contextualization, let us see how classes are constructed. Let c be a class, defined in a subset of D, denoted by D_C. A class c is denoted by $[c_{D_1}, ..., c_{D_i}, ..., c_{D_k}]$, where c_{D_i} is the value of c for the dimension D_i, $D_i \in D_C$. We use a *joker value*, denoted by $*$, which can substitute any value on each dimension in D_C. In other words, $\forall A \in D$, $\forall a \in Dim(A)$, $\{a\} \subset *$.

Thus, an episode e belongs to a class c if the restriction of e on D_C[4] is included in c:

$$\forall A \in D_C,\ e_A \subseteq c_A.$$

[3] Here, a route is different from a journey. The route of a journey is the itinerary followed by the train during this journey. Therefore, several journeys may be associated with a single route (e.g., Paris-Montpellier).

[4] The restriction of e in D_C is the description of e, limited to the dimensions of D_C.

As the environmental and structural criteria are semantically very different, we have identified two structures to represent them: a lattice for environmental classes and a tree for structural classes. We now explain how these structures are defined.

Environmental Lattice. The set of environmental classes can be represented in a multidimensional space containing all the combinations of different environmental criteria as well as their possible values. Environmental classes are defined in the set of environmental dimensions denoted by D_E.

Class	Duration	Exterior Temperature
[*,*]	*	*
[low,*]	low	*
[high,*]	high	*
[low,high]	low	high
[*,high]	*	high
...	...	...

Fig. 2. Extract from environmental classes.

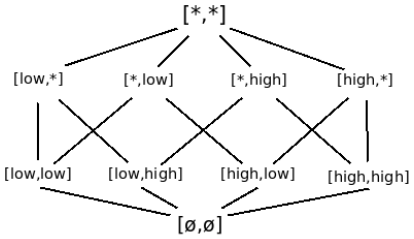

Fig. 3. Environmental Lattice.

Example 2 *Figure 2 shows some of the environmental classes corresponding to the dataset presented in Table 2.*

A class c is denoted by $[c_{ExtT}, c_{Dur}]$*, where ExtT stands for the dimension Exterior Temperature and Dur for the dimension Duration. For example, the class denoted by* $[low, *]$ *is equivalent to the context where the temperature is low (i.e.,* $c_{ExtT} = low$*), for any duration (i.e.,* $c_{Dur} = *$*).*

Using the dataset of Table 2, we can see that the set of episodes belonging to the class $[low, *]$ *is* $\{e_1, e_2, e_4\}$*. Similarly, all the episodes belonging to the class* $[low, high]$ *is* $\{e_1\}$*.*

Environmental classes and their relationships can be represented as a lattice. Nevertheless we first have to define a generalization/specialization order on the set of environmental classes.

Definition 1 *Let* c, c' *be two classes.* $c \geq c' \Leftrightarrow \forall A \in D, v_A \subset u_A$*. If* $c \geq c'$*, then c is said to be more general than* c'*.*

In order to construct classes, we provide a *sum operator* (denoted by $+$) and a *product operator* (denoted by $\bullet$).

The sum of two classes gives us the most specific class generalizing them. The **sum operator** is defined as follows.

Definition 2 *Let c,* c' *be two classes.*

$$t = c + c' \Leftrightarrow \forall A \in D,\ t_A = \begin{cases} c_A\ if\ c_A = c'_A, \\ *\ elsewhere. \end{cases}$$

The product of two classes gives the most general class specializing them. The **product operator** is defined as follows.

Definition 3 *Let c, c' be two classes. Class z is defined as follows: $\forall A \in D, z_A = c_A \cap c'_A$. Then,*

$$t = c \bullet c' \Leftrightarrow \begin{cases} t = z\ if A \in D \mid z_A = \emptyset \\ < \emptyset, ..., \emptyset >\ \ elsewhere. \end{cases}$$

We can now define a lattice, by using the generalization/specialization order between classes and the operators defined above. The ordered set $\langle CS, \geq \rangle$ is a lattice denoted as CL, in which Meet ($\bigwedge$) and Join ($\bigvee$) elements are given by:

1. $\forall T \subset CL, \bigwedge T = +_{t \in T} t$
2. $\forall T \subset CL, \bigvee T = \bullet_{t \in T} t$

Figure 3 illustrates the lattice of environmental classes of the dataset provided in Table 2.

Structural Hierarchy. Structural hierarchy is used to take into account information that could be lost by manipulating episodes. Indeed, it is important to consider the total journey including an episode, and the place of this episode in the journey. Some classes are presented in Figure 4.

id	Route	Index	Fragment
[*]	*	*	*
[J1]	J1	*	*
[J2]	J2	*	*
[J1,E1]	J1	E1	*
[J1,E1,begin]	J1	E1	begin
[J1,E1,middle]	J1	E1	middle
...	...	...	...

Fig. 4. Extract from structural classes.

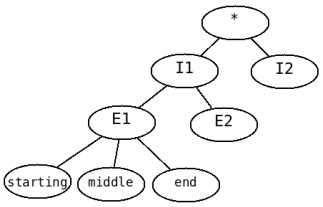

Fig. 5. Structural Hierarchy.

Example 3 *Using the dataset of Table 2, we can see that the set of episodes belonging to the class $[J1]$ is $\{e_1, e_2, e_4, e_5\}$. Similarly, the set of episodes belonging to the class $[J1, E1]$ is $\{e_1, e_4\}$.*

Therefore, we create the hierarchy described in Figure 5, such as the higher the depth of a node is, the more specific is the symbolized class.

Structural classes, defined in D_S, are represented with a tree. The branches of the tree are symbolizing a relationship "is included". A tree is particularly appropriated here because it represents the different granularity levels defined earlier, i.e., from the most general to the finer level: journeys, episodes and fragments.

Let C be the set of all structural classes, and H^C the set of classes relationships. $H^C \subseteq C \times C$, and $(c_1, c_2) \in H^C$ means that c_2 is a subclass of c_1.

A lattice can not represent a structural hierarchy as elements are included in only one element of higher granularity: a fragment belongs to a single episode and an episode belongs to a single route.

- The root of this hierarchy (denoted as $*$) is the most general class, i.e., it contains all the episodes stored in the database.

- Nodes of depth 1 correspond to the various routes made by trains. Thus, the class represented by the node $[J_1]$ contains all the episodes made in the route denoted by J_1.

- The next level takes into account the place of the episode in the path represented by the father node. The class $[J_1, E_1]$ contains all the first episodes of the journey J_1, i.e., episodes of the journey J_1 whose index is 1.

- So far, the classes we have defined contain episodes. However, we have seen that it is possible to use a finer granularity in the data (see Section 2). Indeed, an episode can be divided into three fragments: a starting fragment, a traveling fragment and an ending fragment. To make the most of this level of granularity and obtain more detailed knowledge, we consider classes of fragments in the leaves of the hierarchy. Thus, class $[T_1, E_1, \textit{starting}]$ contains *starting fragments* of the episodes contained in $[T_1, E_1]$. The extraction of frequent sequences in this class will provide knowledge about the behavior at the start of an episode, under the conditions of the class $[T_1, E_1]$.

With the environmental lattice and structural tree presented in this section, we can index knowledge from normal behavior according to their context. Thus, when new data are tested to detect potential problems, we can precisely evaluate the similarity of each part of these new data with comparable elements stored in our normal behavior database.

3.3 Behavior Specificity

In the previous section, we showed how to extract frequent behaviors in classes. To go further, we want to know the specific behavior of each class (i.e., of each particular context). We therefore distinguish the specific patterns for a class (i.e., not found in class brothers in a hierarchy), and the general patterns appearing in several brother classes. General patterns possibly may be specific to a higher level of a class hierarchy.

Using this data representation, we can define the following concepts. Let D be a sequence database, s a sequence, c a class in the structural hierarchy or the environmental hierarchy described in Section 3.2.

Notation 1 *The number of sequences (i.e., episodes or fragments according to the chosen level of granularity) contained in c is denoted as* $nbSeq(c)$.

Notation 2 *The number of sequences contained in c supporting s is denoted as* $suppSeq_c(s)$.

Definition 4 *The support of s in c is denoted as* $supp_c(s)$ *and is defined by:*

$$supp_c(s) = \frac{suppSeq_c(s)}{nbSeq(c)}.$$

Let minSupp be a minimum support threshold, s is said to be frequent in c if $supp_c(s) \geq minSupp$.

Definition 5 *s is a* ***specific pattern*** *in c if:*

- *s is frequent in c,*
- *s is non-frequent in the classes* c' *(where* c' *is a brother of c in the classes hierarchy).*

The notion of specificity provides additional information to the experts. A pattern specific to a class describes a behavior that is linked to a specific context. In particular, this information can be used in order to detect anomalies. For example, if we test the episodes being in $[J_1, E_1, starting]$ and that we meet a lot of behaviors specific to another class, then we can think that these behaviors are not normal in this context and we can investigate the causes of this anomaly more efficiently.

4 Anomaly Detection

In this section, we present how anomaly detection is performed. We consider that we are provided with both one database containing normal behavior on which knowledge have been extracted (see Section 3) and data corresponding to one journey.

The main idea is organized as follows. First, we define a measure to evaluate the compliance of a new journey in a given contextual class. In case of any detected anomaly, we make use of the class hierarchy to provide more detailed information about the problem.

4.1 Detection of Abnormal Behaviors

This involves processing a score to quantify if the sequence corresponding to new data to be tested is consistent with its associated class. We consider that the consistency of an episode with a class c depends on the number of patterns of c

such as the episode is included in. The most the episode is included in patterns of c, the most this episode is consistent with c. We thus introduce a similarity score called the *Conformity Score* which is defined as follows.

Definition 6 *Let s be a sequence to be evaluated in a class c. We denoted by P the set of patterns of c, and P_{incl} the set of patterns of c being included in s. So, the Conformity Score of s, denoted by $conformity(s)$, is such as:*

$$conformity(s) = \frac{|P_{incl}|}{|P|}.$$

We will thereafter denote by $score_c(e)$ the score of an episode e in class c.

4.2 Anomaly Diagnosis

To provide further information on the causes of detected anomalies, we use hierarchies developed in Section 3.2. The detection is performed as follows.

First, we measure the conformity score of a new episode e in the most precise level of each of the hierarchies (i.e., structural and environmental). If the score is high enough in the two classes then the behavior of the train during e is considered as normal. Otherwise, it is possible to distinguish more clearly what is the cause of the anomaly. For example, the anomaly may have structural or environmental reasons.

To obtain more information about the detected anomaly, it is possible to go up in the hierarchy and test the consistency score of e in "parent" classes of problematic classes. For example, if the episode e has a poor score in the class $[low, high]$, then we evaluate its score in the classes $c_1 = [low, *]$ and $c_2 = [*, high]$. If $score_{c_1}(e)$ is inadequate and $score_{c_2}(e)$ is sufficient, then the provided information is that e is in compliance with the normal behaviors related to the exterior temperature, but not with those related to travel duration.

By defining a minimum conformity score *minCo* we can also determine whether the emission of an alarm is necessary or not. Thus, if score $score_c(e)$, then the episode is seen as problematic, and an alarm is emitted. However, we can balance an alarm in relation to the seriousness of the detected problem. For example, a score of *0.1* probably corresponds to a more important issue than a score of *0.45*.

5 Experiments

In order to evaluate our proposal, several experiments were conducted on real datasets. They correspond to the railway data collected on 12 trains where each train has 249 sensors. Each value is collected every five minutes. 232 temperature sensors and 16 acceleration sensors are distributed on the different components (e.g., wheels, motors, etc..) and a sensor measures the overall speed of the train.

5.1 Experimental Protocol

The experimental protocol follows the organization of the proposals:

1. **Characterization of Normal Behavior.**(see Section 3) We have studied the impact of contextualization on the characterization of normal behavior, and the benefit of the search for specific patterns in various contexts.
2. **Anomaly Detection.** (see Section 4) We have evaluated the *conformity score* by applying it on normal and abnormal behavior.

5.2 Normal Behavior Characterization

The discovery of sequential patterns has been performed with the PSP algorithm, described in [13]. We have used a C++ implementation, which can manage time constraints.

Class	Frequent Sequences	Specific Patterns
[*,*]	387	387
[low,*]	876	634
[high,*]	616	411
[low,high]	6430	5859

Table 3. Number of frequent sequences and specific patterns, according to the environmental class.

Table 3 shows, for an extract from the hierarchy presented in Section 3.2, the number of frequent sequences found in each class, and the corresponding number of specific sequences, extracted with a minimum support set to *0.3*. We can note that filtering specific patterns reduces the amount of stored results. Moreover, the fact that each class, including most restrictive ones (i.e., the leaves of the hierarchy), contain specific patterns shows both the importance of the context in railway behavior, and the usefulness of our approach.

Note that for the most general class, denoted as $[*, *]$, the number of specific patterns and frequent sequences is unchanged. This is because this class does not have a brother in the environmental hierarchy. Moreover, we notice in these results that the more a class is specific, the more it contains frequent sequences and specific patterns. Indeed, we look for frequent behaviors. Train behavior heavily depend on surrounding context. Therefore, the more a class is general, the less behaviors are frequent, as they vary much more from one journey to another.

5.3 Anomaly Detection

We have noted in the previous section that we can extract very precise knowledge about normal train behavior. This knowledge is now used in an anomaly detection process, through the methods described in Section 4.

To validate our approach, we use normal behavior from real data and we ensure that they do not generate anomalies. To this end, we have segmented our real data set into two subsets: *(i)* data on which we perform the characterization of normal behavior; *(ii)* data to test.

Figure 6 shows the average score of episodes, depending on the class in the environmental hierarchy. In each class, we have tested 15 randomly selected episodes. Calculated scores are high, in particular in the most specific classes. The average Conformity Score in the most general class (i.e., $[*, *]$) is lower. Indeed, behavior in this class are more various and, given the minimum support used ($minSupp = 0.3$), many frequent sequences appear in a minority of the data.

We have also measured the conformity of a random selection of 15 episodes of the class $[high, *]$ with the normal behavior of the class $[low, high]$. The average score decreases to 0.42, confirming that the behavior of a train when the outside temperature is high does not correspond to what is expected when the temperature is low.

Class	Average Score
[*,*]	0.64
[low,*]	0.85
[high,*]	0.74
[low,high]	0.92

Fig. 6. Average score of tested episodes.

Class	Score
[low,high]	0.05
[low,*]	0.17
[*,high]	0.41
[*,*]	0.3

Fig. 7. Conformity Score of a degraded episode.

Then, in order to detect anomalies in really abnormal behavior, we simulate defects. Indeed, the available real dataset does not contain enough abnormal behaviors to perform valid experiments. Simulated data are created by degrading data in the test dataset. For example, we apply our methods on an episode created by increasing all wheel temperatures by 15^oC. Figure 7 presents the conformity score of this episode in each environmental class which it belongs, from general to specific. The true score of this episode before being corrupted, in its more specific environmental class (e.g., $[low, high]$) is 0.94. However, once the episode is degraded by increasing all wheel temperatures by 15^oC, the conformity score becomes *0.05*. By studying the previous level of the environmental lattice, we can note a difference between the score of the episode in class $[low, *]$ (i.e., low exterior temperature) and in class $[*, high]$ (i.e., long episode duration). The score is higher in the class $[*, high]$. This can be explained by the fact that such heating of wheels sometimes occurs when a train was traveling during a long time, and is not totally unusual in this context. However, this is not the case in the context of the class $[low, *]$.

6 Conclusion

In this paper, we have proposed a new strategy for detecting anomalies from sensor data in the context of railway maintenance. We have addressed the problem of characterizing train behavior with sequential pattern mining. First, we extract some patterns to describe normal behavior and second. Then, we use the previous characterization to determine if a new behavior is normal or not. We are thus able to automatically trigger some alarms.

Our contribution is twofold: *(i)* as the behavior of trains depends on environmental conditions, we have proposed a method of characterization, which originally focuses on contextual knowledge. *(ii)* to detect anomalies, we have developed a conformity measure of new data and when an anomaly is detected we are able to provide information on the seriousness and possible causes of a deviation. In order to validate our approach, we have applied it to real railway data.

This preliminary work opens up interesting prospects. How to make extracted knowledge accessible for the experts (pattern visualization)? How to provide more precise information about detected anomalies? How to apply these methods in a real-time context?

References

1. Rodrigues, P.P., Gama, J.: Online prediction of streaming sensor data. In Joo Gama, J. Roure, J.S.A.R., ed.: Proceedings of the 3rd International Workshop on Knowledge Discovery from Data Streams (IWKDDS 2006), in conjuntion with the 23rd International Conference on Machine Learning. (2006)
2. Yairi, T., Kato, Y., Hori, K.: Fault detection by mining association rules from house-keeping data. In: Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space. (2001)
3. Halatchev, M., Gruenwald, L.: Estimating missing values in related sensor data streams. In Haritsa, J.R., Vijayaraman, T.M., eds.: Proceedings of the 11th International Conference on Management of Data (COMAD '05), Computer Society of India (January 2005) 83–94
4. Chong, S.K., Krishnaswamy, S., Loke, S.W., Gaben, M.M.: Using association rules for energy conservation in wireless sensor networks. In: SAC '08: Proceedings of the 2008 ACM symposium on Applied computing, New York, NY, USA, ACM (2008) 971–975
5. Ma, X., Yang, D., Tang, S., Luo, Q., Zhang, D., Li, S.: Online mining in sensor networks. In Jin, H., Gao, G.R., Xu, Z., Chen, H., eds.: NPC. Volume 3222 of Lecture Notes in Computer Science., Springer (2004) 544–550
6. Guralnik, V., Haigh, K.Z.: Learning models of human behaviour with sequential patterns. In: Proceedings of the AAAI-02 workshop "Automation as Caregiver". (2002) 24–30 AAAI Technical Report WS-02-02.
7. Cook, D.J., Youngblood, M., Heierman, III, E.O., Gopalratnam, K., Rao, S., Litvin, A., Khawaja, F.: Mavhome: An agent-based smart home. In: PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, Washington, DC, USA, IEEE Computer Society (2003) 521

8. Wu, P.H., Peng, W.C., Chen, M.S.: Mining sequential alarm patterns in a telecommunication database. In: DBTel '01: Proceedings of the VLDB 2001 International Workshop on Databases in Telecommunications II, London, UK, Springer-Verlag (2001) 37–51
9. Agrawal, R., Srikant, R.: Mining sequential patterns. In Yu, P.S., Chen, A.S.P., eds.: Eleventh International Conference on Data Engineering, Taipei, Taiwan, IEEE Computer Society Press (1995) 3–14
10. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. SIGMOD Rec. **22**(2) (1993) pp. 207–216
11. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In Apers, P.M.G., Bouzeghoub, M., Gardarin, G., eds.: Proc. 5th Int. Conf. Extending Database Technology, EDBT. Volume 1057., Springer-Verlag (25–29 1996) 3–17
12. Masseglia, F., Poncelet, P., Teisseire, M.: Efficient mining of sequential patterns with time constraints: Reducing the combinations. Expert Systems with Applications **36**(2, Part 2) (2009) 2677 – 2690
13. Masseglia, F., Cathala, F., Poncelet, P.: The psp approach for mining sequential patterns. In Zytkow, J.M., Quafafou, M., eds.: PKDD. Volume 1510 of Lecture Notes in Computer Science., Springer (1998) 176–184