Fuzzy Anomaly Detection in Monitoring Sensor Data

Julien Rabatel, Sandra Bringay, Pascal Poncelet

Fuzzy Anomaly Detection in Monitoring Sensor Data

Julien Rabatel, Sandra Bringay and Pascal Poncelet

Abstract— **Today, many industrial companies must face challenges raised by maintenance. In particular, the anomaly detection problem is probably one of the most investigated. In this paper we address anomaly detection in new train data by comparing them to a source of normal train behavior knowledge, expressed as sequential patterns. To this end, fuzzy logic allows our approach to be both finer and easier to interpret for experts. In order to show the quality of our approach, experiments have been conducted on real and simulated anomalies.**

I. Introduction

Today, many industrial companies must face problems raised by maintenance. The most common solution is called curative maintenance, i.e., equipment is replaced or repaired after the appearance of obvious failures, once the damage has occurred. This maintenance strategy is not adequate for various reasons. Concerning the financial aspect, few hours of downtime can result in millions of dollars in losses for industrial companies. In addition, curative maintenance is also a problem for security reasons. In many critical areas, equipment failures can cause death. For example, it is estimated that approximately 5% of motor vehicle accidents are caused by equipment malfunction or a lack of maintenance[1]. Another aspect is related to the environment and energy saving. Indeed, equipment that is worn or subject to malfunctions often consumes more energy than equipment operating optimally. In addition, systematic maintenance planning is not a satisfactory solution being too expensive compared to real needs. Reducing the problem of equipment maintenance and making maintenance both faster and more effective by anticipating serious breakdowns is thus a very challenging and critical issue.

Preventive maintenance consists in detecting anomalous behavior in order to prevent further damages and avoid more costly maintenance operations. To this end, it is necessary to monitor working equipment. Usually, monitoring data is available through embedded sensors and provides important information such as temperature, humidity rate, etc.

In this paper, we focus on the specific case of train maintenance. Trains monitoring is ensured by sensors positioned on the main components to provide various information. We propose a method to exploit this information in order to assist the development of an effective preventive maintenance.

The requirements in the context of train maintenance are twofold. First, it is important to provide a better understanding of monitored systems. Indeed, since train systems are often complex and contain many components, experts have little knowledge about their behavior. This lack of knowledge makes the problem of maintenance very difficult. Second, it could be of interest to get an overview of the normal behavior. To this end, it is necessary to propose a way for characterizing such normal behaviors from a huge amount of historical data. In this paper, we show how we can be directly address this problem by data mining techniques ([HK06]) associated with fuzzy logic ([Zad65]). We also describe how we design a system for detecting anomalies in train behavior and help experts so that a detected problem can be dealt as promptly as possible.

For example, considering normal data behavior expressed by sequential patterns ([RBP09]), a pattern could be *"In 80% of train journeys, a low value measured by sensor A is followed by an average value measured by sensor B"*, denoted as $p = \langle (A_{low})(B_{avg}) \rangle$. As a consequence, if sensor B is measuring an average value during a new train journey, and we observe the previous value measured by sensor A was low, then we can deduce that the pattern p is concordant with the actual behavior of B. On the other hand, a pattern $\langle (A_{low})(B_{low}) \rangle$ could be considered discordant since it expresses that a low value measured by A is frequently followed by a low value measured by B.

The general problem consists in making a coherent evaluation of the conformity of a sensor behavior with respect to the overall set of concordant or discordant patterns available. In this process, fuzzy logic help refining the representation of concordance and discordance notions. We take into account the fact that a sequential pattern can be more or less concordant or discordant. This aspect allows to emphasize the more relevant patterns in order to enhance the anomaly detection task.

This paper is organized as follows. Section II describes the data representation in the context of train maintenance. We present the anomaly detection for preventive maintenance approach in Section III. Experiments conducted with real and simulated data are described in Section IV. Section V concludes the paper.

II. Data Preprocessing

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

The train monitoring system exploited contains a large set of sensors distributed on the main components of each train allowing to collect various information (e.g., temperature). A complete description of these data is available in [CDA09].

Julien Rabatel is with the LIRMM (CNRS UMR 5506), Univ. Montpellier 2 and Fatronik France Tecnalia (email: rabatel@lirmm.fr).

Sandra Bringay is with the LIRMM (CNRS UMR 5506), Univ. Montpellier 2 and Dpt MIAp, Univ. Montpellier 3, France (email: bringay@lirmm.fr)

Pascal Poncelet is with the LIRMM (CNRS UMR 5506), Univ. Montpellier 2, France (email: poncelet@lirmm.fr).

[1] http://www.smartmotorist.com

TIME	A	B	C	...
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 I

EXTRACT FROM SENSOR DATA.

A. Sensor Data for Train Maintenance

In train maintenance following data must be considered.

Sensors. Each sensor (e.g., A, B, C in Table I) 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. Note that numerical values collected by the sensors are discretized to obtain a set of data more suitable for data mining.

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

We consider handled data as those described in Table I, where a *reading* for a given date (first column) is described by *sensor measurements* (cells of other columns).

The behavior of a sensor is described by the numerical values it measures. The discretization of these sensor values is a preprocessing that, for example, has been used in [HG05] and [CKLG08] to group values expressing the same information into classes of values (e.g., a value could be *low*, *medium* or *high*). For example, temperature values $25℃$ and $26℃$ are different but very close, so they can be regarded as carrying the same information. Moreover, for velocity as well as acceleration sensors, we also consider a zero value because it is a particular value that should not be confused with non-zero values. A zero velocity indicates that the train is stationary, while a non-zero velocity, no matter how low it is, concerns a train in motion.

B. Granularity in Railway Data

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

1) Journey: The definition of a journey is associated to the railway context. For a train, a journey stands for the list of readings collected during the time interval between departure and 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 $minDuration$ be a minimum duration threshold, $maxStop$ a maximum stop duration, and J 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) $\nexists (r_u, ..., r_v, ...r_w) \subseteq J$ such that:

$$\begin{cases}(w-u) > maxStop, \\ and\ \forall\ v \in [u, w],\ velocity(v) = 0.\end{cases}$$

2) Episode: The main issue for characterizing train behavior is to compare similar elements. However, as trains can have different routes the notion of journey is not sufficient (for instance, between two different journeys, we could have different numbers of stops as well as a different delay between two railway stations). This is the reason why we segment the journeys into episodes to obtain a finer level of granularity. Episodes are obtained by relying 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 $velocity(n) = 0$ [2],
- if $m < i < n$, $velocity(i) \neq 0$.

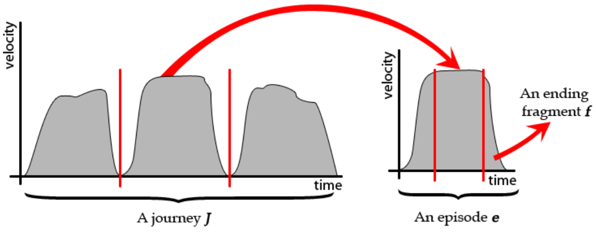

Fig. 1. Segmentation of a journey into episodes.

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.

3) Episode Fragment: The level of granularity corresponding to the fragments is based on the fact that a train's behavior during an episode can be easily divided into 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 $startingfragment$ $E_{starting} = (r_m, ...r_k)$ 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 $endingfragment$

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

$E_{ending} = (r_k, ...r_n)$ 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.

III. 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 is extracted and data corresponding to one new journey.

A. Sequential patterns extraction

In order to characterize normal behavior, we extract in the stored data sequential patterns, introduced in [AS95]. Sequential patterns can be considered as an extension of association rules [AIS93] by handling timestamps associated to items. In the context of sensor data for train maintenance, this problem is adapted as follows.

Given a set of distinct attributes, an *item*, denoted as $i = A_v$, is a sensor A associated to its discretized collected value v at a given time. 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 $s \sqsubseteq s'$, and s' supports 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 greater than or equal to a threshold minimum support ($minSupp$) specified by the user.

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

However, to extract interesting patterns in a database of behavioral sequences, it is important to note that a frequent sequence is interesting only if the gap between each itemset is limited. By extracting frequent sequences in a database of behavioral sequences, we want to highlight the frequent interactions and correlations between sensors, but also between readings. However, the interest of those patterns is strongly associated with the temporal gap between each pair of itemsets in a sequence. To take this into consideration, we modify the concept of subsequence in order to consider only the consecutive itemsets in a sequence.

Definition 1: A sequence s is a *contiguous subsequence* of the sequence s', denoted as $s \sqsubseteq^c s'$, if there exist three sequences s_1, s_2, and s_3 such that $s' = s_1 + s_2 + s_3$, $|s| = |s_2|$, and $s \sqsubseteq s'$.

Example 1: The sequence $\langle (A_{avg})(B_{avg}) \rangle$ is a contiguous subsequence of $\langle (A_{low})(A_{avg})(A_{high} B_{avg}) \rangle$, but $\langle (A_{low})(B_{avg}) \rangle$ is not.

In order to use a relevant set of sequential patterns, we also have to consider the fact that handled data are highly redundant. Indeed, the behavior described by sensor measurements are generally stable for a reading to another. This characteristic is particularly visible for data that change slowly over time (e.g., a wheel temperature). Therefore, we are dealing with very large sequences with consecutive itemsets containing redundant information. Such sequences bring two problems: *(i)* the mining of such sequences is particularly difficult (the repetition of identical itemsets considerably increases the search space) and *(ii)* it yields no additional information. We therefore propose the concept of aggregated sequence.

Definition 2: Let $s = \langle I_1 I_2 ... I_n \rangle$ be a sequence. The corresponding *aggregated pattern*, denoted by s^*, is the maximal subsequence of s respecting the following condition:

$s^* = \langle I_1^* I_2^* ... I_i^* ... I_m^* \rangle$, such that $\forall I_i^* \in s^*, I_i^* \neq I_{i+1}^*$.

Note that $s^* \sqsubseteq s$.

Example 2: Let $s = \langle (A_{low} B_{avg})(A_{low})(A_{low})(B_{high}) \rangle$. The aggregated pattern of s, denoted by s^*, is:

$$s^* = \langle (A_{low} B_{avg})(A_{low})(B_{high}) \rangle.$$

To adapt the problem of mining sequential patterns in the context of train monitoring data, we only consider the contiguous aggregated sequential patterns. Given previous definitions, we can obtain patterns such as $p = \langle (A_{low})(B_{avg}) \rangle$ with $support(p) = 80\%$, meaning that *"in 80% of train journeys, a low value measured by sensor A is followed by an average value measured by sensor B"*.

This demonstrates the particular use of sequential patterns for addressing maintenance problems. Easily interpreted (as shown above), sequential patterns provide experts with better understanding of normal behavior and detected anomalies.

B. Preliminary Definitions

During a journey, each component (and each sensor) has a behavior which is related to the behavior of other components. These interactions, both temporal and inter-components, are described by a set of extracted sequential patterns. Therefore, we want to make an evaluation of the behavior of a component based on this obtained knowledge. The main idea is relying on the following remarks:

- we can consider that a sensor behaves normally if we find enough patterns in our knowledge base that validate its current state,
- we can say that its behavior is anomalous if we find enough patterns that contradict it,

Thus, for each reading and each sensor, we compute two scores: a concordance score and a discordance score. Depending on the value of these scores, we can then indicate whether the behavior is normal, anomalous, or uncertain. Below, we describe how the various needed scores are calculated.

In order to use the extracted sequential patterns for measuring the compliance of a sequence describing a new journey, first of all we introduce the concept of covering sequence.

Definition 3: Let I be an itemset and $s = \langle I_1 ... I_i ... I_n \rangle$ a sequence. I is *covering* s if $\forall I_i \in s$, $I \subseteq I_i$.

Example 3: The itemset (A_{low}) is covering the sequence $\langle (A_{low} B_{low})(A_{low}) \rangle$.

Definition 4: Let $p = \langle I_1^* ... I_i^* ... I_l^* \rangle$ be an aggregated pattern and $s = \langle I_1 ... I_j ... I_m \rangle$ a sequence. p is *covering* s, denoted by $p \prec s$, if there exists a set of sequences $\{s_1, s_2, ..., s_m\}$ such that:

i) $s = s_1 + s_2 + ... + s_m$,

ii) $\forall i | 1 \leq i \leq m$, I_i^* is covering I_i. Moreover, I_i^* is called the corresponding itemset of I_i in p.

Example 4: Let s be a sequence, and p an aggregated pattern, such that:

$$s = \langle \overbrace{(A_{low})(A_{low} B_{low})}^{s_1} \overbrace{(A_{avg} B_{avg})(A_{avg} B_{avg})}^{s_2} \overbrace{(B_{high})}^{s_3} \rangle$$
$$p = \langle \underbrace{(A_{low})}_{I_1} \underbrace{(A_{avg} B_{avg})}_{I_2} \underbrace{(B_{high})}_{I_3} \rangle.$$

We can note that s can be broken down into 3 sequences s_1, s_2 and s_3, such that $I_1 \sqsubseteq s_1$, $I_2 \sqsubseteq s_2$, and $I_3 \sqsubseteq s_3$. Thus, $p \prec s$.

On the other hand, p is not covering the sequence

$$s' = \langle (A_{low})(A_{low} B_{low})(A_{avg})(A_{avg} B_{avg})(B_{high}) \rangle$$

.

Using the notion of covering sequence, we can now describe two types of patterns: *(1)* concordant patterns, validating the behavior of a sensor at a given time, and *(2)* discordant patterns contradicting this behavior.

Definition 5: Let $A \in \mathbf{\Omega}$, $s = \langle I_1 I_2 ... I_n \rangle$ a sequence, and $p = \langle I_1^* I_2^* ... I_m^* \rangle$ an aggregated pattern. p is a concordant pattern for A in the i^{th} itemset of s, i.e., a (A, i)-*concordant pattern* in s, if:

i) there exist integers h, j such that $1 \leq h \leq i \leq j \leq n$, and $p \prec \langle I_h ... I_i ... I_j \rangle$.

ii) let I^* be the corresponding itemset of I_i in p, there exists an item $A_v \in I^*$.

Example 5: Let A, B and C be sensors, $p_1 = \langle (A_{avg} B_{low})(Bavg) \rangle$, $p_2 = \langle (A_{avg}) \rangle$ and $p_3 = \langle (A_{low})(A_{avg}) \rangle$ three aggregated patterns, and s a sequence such that:

$$s = \langle (A_{avg} B_{low})(A_{high} B_{avg})(A_{high} B_{high}) \rangle.$$

The aggregated patterns p_1 and p_2 are $(A, 1)$-*concordant patterns*. On the other hand, p_3 is not a $(A, 1)$-*concordant pattern*.

A discordant pattern for the state of a sensor at a given time describes an unexpected behavior.

Definition 6: Let $A \in \mathbf{\Omega}$, $s = \langle I_1 ... I_i ... I_n \rangle$ a sequence such that I_i contains an *A-item*, denoted by i_s, and $p = \langle I_1^* I_2^* ... I_j^* ... I_m^* \rangle$ an aggregated pattern such that I_j^* contains an *A-item*, denoted by i_p. p' is the sequence p where i_p is replaced by i_s in I_j.
p is a discordant pattern for A in the i^{th} itemset of s, i.e., a (A, i)-*discordant pattern* in s if:

i) p is not a (A, i)-*concordant pattern*,

ii) p' is a (A, i)-*concordant pattern.*

The items i_s and i_p are called *discordant items*. More precisely, i_s is the discordant item of s, and i_p is the discordant item of p.

Example 6: Let A, B and C be sensors, $p_1 = \langle (A_{low} B_{low})(B_{avg}) \rangle$ and $p_2 = \langle (A_{high} B_{avg}) \rangle$ two aggregated patterns, and s a sequence such that:

$$s = \langle (A_{avg} B_{low})(A_{high} B_{avg})(A_{high} B_{high}) \rangle.$$

The aggregated pattern p_1 is a $(A, 1)$-discordant pattern. On the other hand, p_2 is not a $(A, 1)$-discordant pattern.

C. Conformity Score

Preliminary definitions give us the possibility to identify, for a given sensor A in the i^{th} itemset of a given sequence s, the set of *(A,i)-concordant patterns* denoted as $\mathcal{P}_c$ and the set of *(A,i)-discordant patterns* denoted as $\mathcal{P}_d$. We will also use $maxSize$ as $max(size(p))$, for $p \in \mathcal{P}_c \cup \mathcal{P}_d$, where $size(p)$ is the number of items in p.

In the following examples, we will consider the sequence s, such that:

$$s = \langle (A_{low} B_{low})(A_{low} B_{avg})(\mathbf{A_{low}} B_{avg})(A_{avg} B_{avg}) \rangle \ ,$$

and the set of aggregated patterns as well as their support contained in Table II.

ID	Aggregated pattern	Support
p_1	$\langle (A_{low})(A_{avg} B_{avg}) \rangle$	25 %
p_2	$\langle (A_{low} B_{avg}) \rangle$	65%
p_3	$\langle (A_{low} B_{avg})(A_{avg}) \rangle$	60%
p_4	$\langle (A_{low} B_{low})(A_{low})(A_{low} B_{avg})(A_{avg}) \rangle$	50%
p_5	$\langle (A_{avg} B_{avg})(B_{avg}) \rangle$	45%
p_6	$\langle (A_{high} B_{avg}) \rangle$	20%
p_7	$\langle (A_{avg} B_{high})(2_{high}) \rangle$	45%

TABLE II

SET OF AGGREGATED PATTERNS.

In order to calculate an overall conformity score, we have to consider all concordant and discordant patterns, because

ID	weight	μ_c	μ_d
p_1	0.75	0.125	0
p_2	1.3	0.21	0
p_3	1.8	0.3	0
p_4	0.5	0.5	0
p_5	-1.35	0	0.075
p_6	-0.2	0	0.033
p_7	0	0	0
SUM ⟶		1.135	0.108

TABLE III
MEMBERSHIP DEGREES OF PATTERNS.

each of them will have an impact on the final score. However, as we will see thereafter, patterns will affect the conformity score differently, depending on specific features. We propose to take into account the specificities of each pattern, by defining a weight for each sequential pattern.

1) Concordant Patterns: All concordant patterns can not be considered as equally relevant. For example in Table II, $p3$ and $p4$ are (A, i)-concordant patterns in s, and have close supports. However, $p4$ can be considered as more relevant than $p3$ for defining the concordance of sensor A according to our extracted knowledge about normal behavior. Indeed, the situation described by $p4$ is more precise than that described by $p3$. This precision is directly related with the size of a concordant pattern.

In addition, by considering the *(A,i)-concordant patterns* $p1$ and $p3$, we can highlight the fact that the support is also important in order to evaluate the confidence we have in a concordant pattern for estimating the conformity of a sensor state. Indeed, $p1$ and $p3$ have the same size, but their support is very different, involving that the situation described by $p3$ has been observed more often than that described by $p1$.

In consequence, the weight associated with concordant pattern depends on its size and its support.

2) Discordant Patterns: In a similar way, discordant patterns have different specific aspects. A discordant pattern is also described by its support and its size. Note that the important part of a discordant pattern is the part covering the tested sequence. Therefore, in the weight of a discordant pattern, we consider its size without the discordant item.

Moreover, to evaluate the weight of a discordant pattern, it is important to consider a discordance degree, describing the gap between this value and the "expected value".

a) Discordance Degree: Let $A \in \mathbf{\Omega}$ a sensor, and $\mathcal{D} = dom(A)$. $\mathcal{D}$ is such that $\mathcal{D} = (v_1, ...v_i, ..., v_n)$, where v_i is a discrete value.

Definition 7: Let p be a (A, i)-discordant pattern in a sequence s, i_p the discordant item of p, and i_s the discordant item of s. We define the *discordance degree* of p as follows. Let us consider $v_k \in \mathcal{D}$ and $v_l \in \mathcal{D}$, such that $v_k = value(i_p)$ and $v_l = value(i_s)$. The discordance degree of p, denoted by $discDegree(p)$, is:

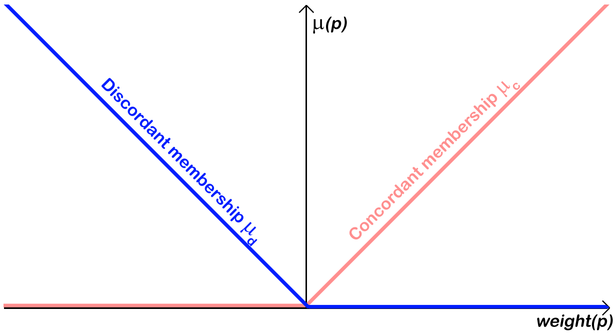

Fig. 2. Fuzzy partition.

$$discDegree(p) = \frac{|l - k|}{n - 1}.$$

Hence, the discordance degree is maximal and equal to 1 when the gap between the actual and the "expected" value is maximal.

Example 7: In the previous sequence s, $dom(A) = dom(B) = (i_1, i_2, i_3)$, such that $i_1 = low$, $i_2 = avg$, and $i_3 = high$. By considering the sequence s and the discordant pattern p_5, we can note that the discordant item of s is A_{low} and the discordant item of p_5 is A_{avg}. Thus, the discordance degree of p_5 is:

$$discDegree(p_5) = \frac{|2 - 1|}{|dom(A)| - 1} = 1/2.$$

We can now define the *weight* of any aggregated pattern.

Definition 8: Let p be an aggregated pattern. The *weight* of such a pattern is defined as follows:

$$weight(p) =$$
$$\begin{cases} ||p|| \times support(p) \ if \ p \in \mathcal{P}_c, \\ -(||p|| - 1) \times support(p) \times discDegree(p) \ if \ p \in \mathcal{P}_d, \\ 0 \ otherwise. \end{cases}$$

Note that the weight of a discordant pattern is negative, to distinguish it from concordant patterns.

3) Fuzzy Partition: In order to consider each pattern in a relevant way, we consider a fuzzy partition in two fuzzy sets *Concordant* and *Discordant* defined over the universe U of the values of pattern weights. The membership functions of these fuzzy sets, respectively denoted as μ_c and μ_d are defined as follows: $\mu_c(p) = \begin{cases} \frac{weight(p)}{maxSize} \ if \ p \in \mathcal{P}_c, \\ 0 \ otherwise. \end{cases}$

and $\mu_d(p) = \begin{cases} \frac{|weight(p)|}{maxSize} \ if \ p \in \mathcal{P}_d, \\ 0 \ otherwise. \end{cases}$

In order to provide us with a general score of conformity, we define at first two scores corresponding to the global score of concordance and the global score of discordance.

Definition 9: The *concordance score* of a sensor A in the i^{th} itemset of a sequence s is: $score_{conc}(A, i) = \sum_{p \in \mathcal{P}^c} \mu_c(p)$.
The *discordance score* of a sensor A in the i^{th} itemset of a sequence s is: $score_{disc}(A, i) = \sum_{p \in \mathcal{P}^d} \mu_d(p)$.

Note that considering a membership degree here allows to provide experts with more interpretable information about relevant patterns (i.e., concordant and discordant patterns).

Example 8: Table III describes, for each pattern presented in Table II, its weight and its membership degree in the fuzzy sets *Concordant* and *Discordant*. For example, $p3$ is a (A,i)-concordant pattern in s. Its weight is 1.8, its membership degree is 0.5 in the *Concordant* fuzzy set and 0 in the *Discordant* fuzzy set.

In addition, this table provide in its last line the concordance score (1.135) and discordance score (0.108).

4) Conformity Score: For each sensor and each itemset in a sequence, we defined how to calculate a concordance score and a discordance score. We can now address the problem of defining a global *conformity score* of a sensor A in the i^{th} itemset of a sequence, denoted as $score(A,i)$. This score, defined between -1 and 1, must meet the following requirements:

- if $score(c,t)$ is close to 1, the state of A in i is considered as normal,
- if $score(c,t)$ is close to -1, the state of A in i is considered as abnormal,
- if $score(c,t)$ is close to 0, the state of A in i is considered as uncertain.

Definition 10: Let $A \in \mathbf{\Omega}$ be a sensor and s a sequence. The *conformity score* of A in the i^{th} itemset of s, denoted by $score(A,i)$ is defined as follows:

$$score(A,i) = \frac{score_{conc}(A,i) - score_{disc}(A,i)}{max(score_{conc}(A,i), score_{disc}(A,i))}.$$

Example 9: By considering the previous examples, we can now calculate the *conformity score* of A in the 3^{rd} itemset of s as follows:

$$score(A,3) = \frac{score_{conc}(A,3) - score_{disc}(A,3)}{max(score_{conc}(A,3), score_{disc}(A,3))} = \frac{1.135 - 0.108}{1.135} = 0.905$$

One important issue when addressing the problem of detecting anomalies concerns false alarms, especially when data are noisy. Thus, it is possible that a sensor has a bad score on a reading, which do not correspond to a real problem. In this case, the score of the sensor then quickly goes back to normal values. To avoid this, it is preferable to take into account the surrounding score values of the same sensor. The smoothed score is thus the average score of a sensor over a specified window.

Definition 11: Let $A \in \mathbf{\Omega}$ be a sensor, s a sequence, and w the smoothing window. The *smoothed conformity score* of A in the i^{th} itemset of s, denoted by $score_w(A,i)$ is defined as follows:

$$score_w(A,i) = \frac{\sum_{j=i-w}^{i+w} score(A,j)}{2w+1}.$$

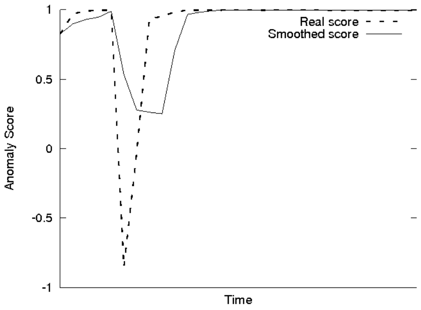

Fig. 3. Influence of the smoothing window on the anomaly score.

Figure 3 shows the evolution of the score of a sensor during a fragment, with and without smoothing (with $w = 3$). Without smoothing, we can see a decrease of the value that could be interpreted as an anomalous behavior. However, the immediate increasing of the score indicates that it is not a real anomaly. The smoothed score is an efficient method for restricting this phenomenon.

We have described throughout this section how to use a list of sequential patterns, illustrating the normal behavior of the monitored system, to detect abnormal behavior among new collected data. This approach is particularly suited to the problem of monitoring complex industrial systems because it does not only declare a data sequence as normal or abnormal, but provides also very precise information about the anomaly detected: the localization of the fault (which sensor, which component, etc..), the precise moment when this anomaly occurred, and a score quantifying the seriousness of the detected problem.

IV. Experimental Results

In order to evaluate our proposal, several experiments were conducted on a real dataset from data described in section II, over a one-year period.

The discovery of frequent sequences has been performed with the PSP algorithm described in [MCP98] with the minimum support set to 0.3.

We described in section III the approach used to detect anomalies in new data. Thus, we can use it to classify new fragments of journeys into two categories: normal data and abnormal data. To evaluate our approach in an appropriate manner, it is necessary to conduct experiments on both types of data. However, if it is easy to find data which do not contain faults, it is often more difficult to obtain a large data set containing anomalies. For this reason, we have simulated a set of anomalous data, on which we have conducted our experiments.

To this end, we have used the real normal data set, and we have corrupted the data in order to reproduce classic behavioral anomalies on the values collected by sensors. Anomalies generated are threefold:

- **Blocked values** (see Figure 4): the value collected by a sensor is not refreshing. This anomaly is usually related to a faulty sensor, or a problem of transmitting information between the sensor and the database, but very rarely describes a real problem on a component.

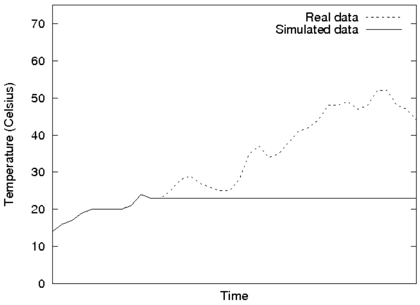

Fig. 4. Simulated anomaly: blocked values.

- **Shifted values** (see Figure 5): a value that is shifted in comparison with the normal behavior of a sensor (or group of sensors): a constant value is added (or substracted) to the real collected value. This type of anomaly may, for example, describe an abnormal overheating of a component.

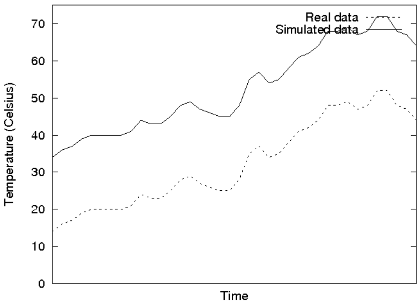

Fig. 5. Simulated anomaly: shifted values.

- **Random values** (see Figure 6): in this case, collected values are randomized. They describe an aberrant behavior without any link with the expected behavior.

We simulated these anomalies in order to build a data set containing about 600 fragments of episodes (i.e., 200 episodes) composed as follows:

- 300 are fragments of real data validated as normal (i.e., without anomalies),

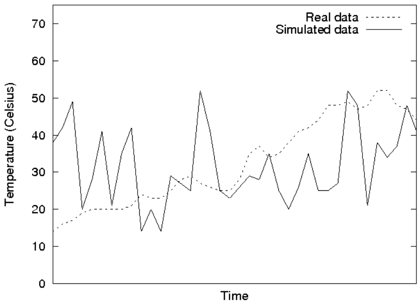

Fig. 6. Simulated anomaly: random values.

- 300 fragments containing anomalies were generated from normal data. The three types of anomalies described above are distributed equally between these fragments.

The approach was tested on the basis of cross-validation, by segmenting the total set of data into 10 equal parts. Thus, each part contains 600 fragments, of which 300 are normal data. Note that the learning step (i.e., characterization of normal behavior) is performed on normal data only. To quantify the number of anomalous fragments in our data set, we consider a fragment abnormal if its anomaly score falls below *-0.5*.

	Predict. Anomalous	Predict. Normal
Real Anomalous	98	2

TABLE V
CONFUSION MATRIX FOR RANDOM VALUES ANOMALIES.

	Predict. Anomalous	Predict. Normal
Real Anomalous	85	15

TABLE VI
CONFUSION MATRIX FOR BLOCKED VALUES ANOMALIES.

	Predict. Anomalous	Predict. Normal
Real Anomalous	92	8

TABLE VII
CONFUSION MATRIX FOR SHIFTED VALUES ANOMALIES.

Thus, we have obtained the confusion matrices presented in Table IV, containing the average results obtained by cross validation on the entire test set. We can note that the system is well balanced and has similar qualities to both recognize the normal behaviors, but also to identify anomalies.

	Predict. Normal	Predict. Anomalous	Recall	Precision
Real Normal	272	28	90.67%	91.58%
Real Anomalous	25	275	91.67%	90.76%
Global results	297	303	**91.17%**	**91.17%**

TABLE IV
GLOBAL CONFUSION MATRIX.

In order to evaluate the results, we have calculated two measures of quality widely used: the precision and the recall. Moreover, recall and precision is in all cases above 90%, so the approach limit both the number of false alarms and the number of undetected anomalies.

Tables V, VI and VII contain the results for each type of anomaly. We note that the results are very good to recognize "*random values*" or "*shifted values*" anomalies. This is due to the fact that the simulated behavior is very different from a normal behavior. In contrast, "*blocked values*" anomalies are different: the anomalous value may not be sufficiently distant from "expected" values for detecting the problem.

V. CONCLUSION

In this paper, we addressed the problem of complex system maintenance. Existing strategies looked at performing curative maintenance, (making the necessary maintenance operations after the occurrence of a failure). This is not suited to this context as too costly and too dangerous. On the other hand, a planned and systematic maintenance is too expensive, although it can usually avoid serious failures. We addressed the problem of developing an approach allowing preventive maintenance, a good compromise between the two previous solutions. Preventive maintenance consists in detecting abnormal behavior early enough for performing the necessary operations.

However, the complexity of such systems (e.g., trains, industrial machinery, etc.) makes their behavior difficult to understand and interpret. In these circumstances it is particularly difficult to detect abnormal behavior that often forsees significant and costly failures.

The problem addressed is particularly challenging since errors in diagnosis may cause numerous inconveniences. We thus developed an approach to use data collected by sensors in order to analyze behaviors and allow the detection of anomalies. Our contribution is divided into three parts. First, we propose an adequate representation of data in order to extract knowledge based on sensor data describing the past normal behavior of systems. Second, we study the possibility to extract sequential patterns in historical data thus improve understanding of systems for experts, but also provide a knowledge database used to develop an anomaly detection method. Finally, we propose an approach to compare new journeys with sequential patterns describing normal behavior. We provide experts with an anomaly score for each sensor and each sensor reading. The approach allows to locate precisely the anomalies and to quantify the extent to which the anomaly seems problematic.

One main advantage of the presented approach is that all obtained results are easily interpretable for decision makers. This is particularly important since users must be able to make informed decisions. This feature is largely provided by the use of sequential patterns easily translatable in natural language, and fuzzy logic for understanding which patterns are really significant for anomaly detection.

Although the presented work meets our expectations in terms of results, it opens interesting perspectives. In particular, the development of a graphical user interface will allow users to access results quickly and efficiently. Another important aspect may be to adapt the approach to a real-time context. This will detect the anomalies on running trains. Furthermore, an interesting question may be further investigated: how to manage the evolution of normal behavior over time? This will conduct the knowledge database to be incrementally updated to ensure knowledge validity over time.

REFERENCES

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2), 1993.

[AS95] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In Philip S. Yu and Arbee S. P. Chen, editors, *Eleventh International Conference on Data Engineering*. IEEE Computer Society Press, 1995.

[CDA09] A. Carrascal, A. Díez, and A. Azpeitia. Unsupervised Methods for Anomalies Detection through Intelligent Monitoring Systems. In *Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems*, page 144. Springer, 2009.

[CKLG08] Suan Khai Chong, Shonali Krishnaswamy, Seng Wai Loke, and Mohamed Medhat Gaben. Using association rules for energy conservation in wireless sensor networks. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 2008.

[HG05] Mihail Halatchev and Le Gruenwald. Estimating missing values in related sensor data streams. In Jayant R. Haritsa and T. M. Vijayaraman, editors, *Proceedings of the 11th International Conference on Management of Data (COMAD '05)*. Computer Society of India, 2005.

[HK06] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.

[MCP98] Florent Masseglia, Fabienne Cathala, and Pascal Poncelet. The psp approach for mining sequential patterns. In Jan M. Zytkow and Mohamed Quafafou, editors, *PKDD*, volume 1510 of *Lecture Notes in Computer Science*. Springer, 1998.

[RBP09] J. Rabatel, S. Bringay, and P. Poncelet. SO_MAD: SensOr Mining for Anomaly Detection in Railway Data. In *Proceedings of the 9th Industrial Conference on Advances in Data Mining. Applications and Theoretical Aspects*, page 205. Springer, 2009.

[Zad65] L.A. Zadeh. Fuzzy sets. *Information Control*, 8:pp. 338–353, 1965.