Scoring of Machine-Learning Algorithms for Providing User Guidance in Special Purpose Machines

Valentin Plenk*, Sascha Lang†, Florian Wogenstein‡
Institute of Information Systems at Hof University, Hof, Germany
Email: *valentin.plenk@iisys.de, †sascha.lang@iisys.de, ‡florian.wogenstein@iisys.de

***Abstract*—We propose a system to make complex production machines more user-friendly by giving the operator recommendations, such as "in the last 10 occurrences of this event the operators performed the following keystrokes". We describe algorithms to generate the recommendations based on data on former user-interaction and process values and to store them in a knowledge base. We also propose algorithms to retrieve recommendations suited to the current process state. We evaluate their performance on simulated data and data gathered from real production machines.**

Keywords–machine-learning; human machine interfaces; special-purpose machines; production machines

I. Introduction

In [1] we proposed to apply machine learning algorithms to generate recommendations like "in the last 10 occurrences of this event the operators performed the following keystrokes" for operators of complex production machines. Figure 1 shows the context: our system, the black box, builds a knowledge base from past operator interactions with a complex production machine.

The system faces two challenges: One is to retrieve the recommendations that are most suitable for the current state of the production machine. The other is to build the knowledge base by extracting the operator interactions from data logged during production.

Despite of the idea being quite straightforward there is apparently little similar work. Most of the work in the production-machine sector applies machine learning to applications dealing with pattern detection in process data or distingushing different datasets [2]–[5]. The algorithms are used to flag errors or problems with production quality to the machine's operator who needs expertise to counter the detected problem.

Challiol et al. [6] describes an application striving to display content dependent on the users context. While addressing a completely different application domain this is similar to our approach in terms of matching context to content. The authors assume the content to be available and do not propose specific context matching algorithms. The context matching algorithm proposed in [7] is quite abstract as is its performance evaluation presented in [8].

Our approach as described in Section II covers aspects similar to [6], [7] and [8] with strong focus on the application in complex production machines. Furthermore we propose to automatically generate the content of the recommendations.

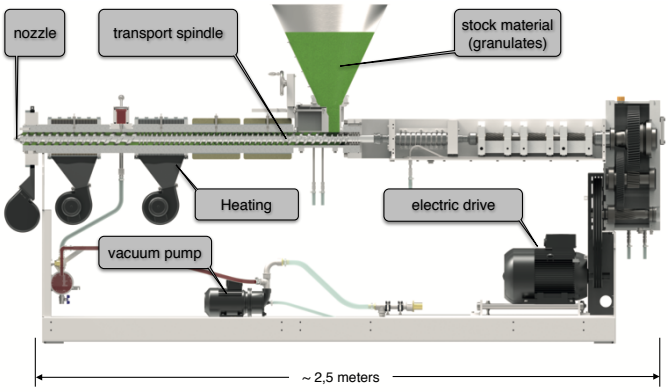
The algorithms presented in Section III build a knowledge base that can be edited. i.e., the recommendations can be presented to an experienced operator who can modify or delete them. This is a marked difference to most machine learning algorithms whose knowledge base cannot be edited.

Section IV evaluates the performance of these algorithms on test data. These data are used to train the algorithms and to derive the "truth" we compare to the generated recommendations.

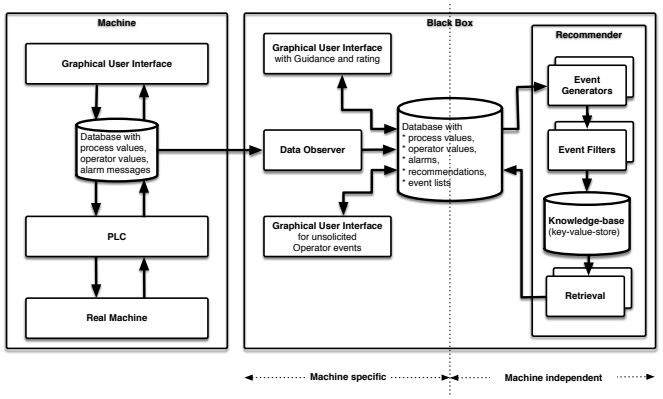
Section VI summarizes the results and gives a short outlook for our future work and the challenges expected in the near future.

II. Proposed Approach

The basic idea of our approach is to extract operator knowledge from the continuous stream of PLC-variables logged during the operation of the machine. We want our algorithms to be as generic as possible and to work without deeper understanding of the machine. However, we need information to recommend actions to the machine operator: Which columns



a) Plastics extruder



b) System architecture

Figure 1. Application context for our machine learning algorithms

 ISBN: 978-1-61208-592-0

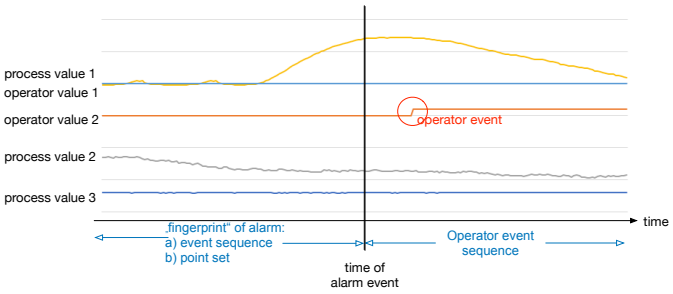


Figure 2. Principle of user guidance generation

in the PLC variable database correspond to values affected by the process, e.g., actual temperature, and which are controlled by the operator, e.g., temperature set-point. We call the first set of variables process values and the second set operator values. The rows of the database hold consecutive values for these variables.

With this information we propose to generate user guidance as shown in Figure 2: The alarm messages in the alarm database are used to tag discrete parts of the endless stream of data logged in the PLC-variable database. The alarm tag and the data before the occurrence of the alarm are regarded as a fingerprint that identifies the alarm. This fingerprint consists of process and operator variables. The data after the occurrence of the alarm also consists of process and operator values. The actions performed by the operator will change some of these operator values. By detecting these changes we build a sequence of operator events. This sequence represents the actions of the operator corresponding to a fingerprint.

With fingerprints and corresponding operator sequences we build a key value store representing the operators' expert knowledge. The fingerprint is the key and the operator sequence the value. These key-value pairs are generated by monitoring the continuous stream of data logged during the operation of the machine. When the machine raises an alarm the algorithms generate a key for that new problem. Now the key-value pairs in the knowledge base are searched for the best matching elements. The best matches are then provided as a recommendation on how to solve the problem. The fingerprint and the operator sequence performed by the operator are then stored in the knowledge base as a new key value pair.

In Section III we describe two generally different approaches for the processing of the fingerprints: One set of algorithms directly uses the process and operator variables. The other set converts the data in the fingerprint into an event sequence. Events are generated for the parts of a time series where the data changes. Several events for one or several columns constitute an event sequence. Detailed descriptions of the event generation can be found in [1] and [9].

III. Recommendation Generation

In case the machine needs operator assistance, i.e., it raised an alarm by adding a new row in the error message table, our guidance generation algorithm is triggered. The fingerprint of the current situation is used as a search key for the knowledge base.

Section III-A describes a set of algorithms that converts the data in the fingerprint into an event sequence and then searches for this sequence. Section III-B introduces the best performing algorithm that directly uses the N-dimensional point set of process and operator variables in the fingerprint as a key. More algorithms are described in [9]

A. Recommendation Generation using Event Sequences

1) Map: In [1] we used a content addressable memory, i.e., a Java map, to store the knowledge base. We build the knowledge base by iterating through all past occurrences of alarms. For each fingerprint we calculate the event sequence and use it as the key and the corresponding operator event sequence as the value. Each key-value pair is then stored in the map. For ambiguous entries we store the frequency of the respective sequence in the past.

To generate a recommendation we simply generate the event sequence corresponding to the fingerprint of the current alarm and query the map for this key. For map entries with several values for one key we return several recommendations and their frequency in the past. Should the key not be in the map, we cannot generate a recommendation.

2) Map with Statistical Event Filtering: The map presented in Section III-A1 will only find a recommendation if the search key exactly matches one of the stored keys. If the event sequences contain spurious events, e.g., caused by noise, we can not find a match. So we created an algorithm to suppress the irrelevant events.

We use a statistical approach identifying unimportant events to remove them from the process event sequence. The filtered event list is then processed as described in Section III-A1.

The filtering is done in two steps. First we iterate through the event sequences of all stored fingerprints for one alarm and count how often an event is contained in the set of event sequences. In the next step we remove all events with a frequency below a defined threshold from the event sequences. So far our tests indicate that 0.5 is a reasonable choice.

3) String matching: The map algorithm presented in Section III-A1 requires the key in the query to be absolutely identical to one key in the map. Thus, it is very sensitive to changes in the key, i.e., the event-sequence. We alleviate this rigorous selection criterion by storing all keys and the corresponding values in a vector. We then loop through all the elements and compare the sequence of the query to the sequence stored in the vector. We return the value as a recommendation that is most similar to the key in the query.

We evaluate the similarity with the Levenshtein distance for string values as described in [10]. For the comparison we treat each event in the sequence as one letter.

B. Recommendation using point sets

The approach described in Section III-A3 allows for disturbance by not requiring a complete match between search key and stored key. We can take this idea one step further by regarding the data points in the fingerprint as a set of n-dimensional points. For one timestamp every process and operator value is one dimension.

As in Section III-A3 we store the point sets and the corresponding operator events in a vector of key-value-pairs. To generate a recommendation we loop through all the elements of this vector and compare the fingerprint, i.e., the point set,

 ISBN: 978-1-61208-592-0

of the query to the point sets stored in the vector. We return the value, i.e., the operator events, as a recommendation that is most similar to the key in the query.

Plenk et al. [9] details several procedures for matching point sets. In this paper we only detail the procedure that performed best on our data. It is a combination of the Hausdorff distance for point sets and the Manhattan-distance for points.

The Hausdorff-distance defines a similarity between two not empty point sets A and B. For one point a that is an element of set A the distance between a and B is defined as:

$$D_H(a, B) = \min_{b \in B} d(a, b) \tag{1}$$

where d is the Manhattan-distance between two points a and b with the dimension i:

$$d_m(a, b) = \sum_i |a_i - b_i| \tag{2}$$

On that basis the the Hausdorff-distance between two sets A and B can be put down as:

$$d_h(A, B) = \max \left\{ \max_{a \in A} D(a, B), \max_{b \in B} D(b, A) \right\} \tag{3}$$

IV. Quality of recommendations

To evaluate our algorithms we use a "batch-mode" that basically iterates through a long time series of PLC-variables and alarms. It triggers the recommender for each alarm encountered during the iterations and requests a recommendation. This recommendation is compared to the operator sequence stored in the time series. This sequence is taken as the expected recommendation or "truth" for this particular alarm. If the recommendation is equal to the operator sequence in the test-set the test is positive and C_{pos} is incremented. If a different sequence is returned or the algorithm returns nothing the test fails and C_{neg} is incremented. With these two parameters we can calculate the Quality Q_{alg} for the algorithm:

$$Q_{alg} = \frac{C_{pos}}{C_{neg} + C_{pos}} \cdot 100\% \tag{4}$$

We need to train the recommender algorithms before feeding them alarms. We generate the training data by dividing the dataset into four subsets with an equal number of alarms. One of these sets is taken as the test-set. The others are combined into a training set. Thus, we can run four tests on each of our algorithms by using the test-sets as training data. The remaining sets of alarms and data are used to train the algorithms.

The expression in equation 4 ranges between $0 \leq Q_{alg} \leq 100\%$. A closer look at the test-sets however reveals cases where an operator sequence only occurs in the test set and not in the training set. Thus, the algorithm cannot learn this recommendation and is not able to detect it correctly. Some other keys have more than one associated value. In these cases, the algorithm is not able to decide which operator sequence is the right one. Consequently the maximum possible score is less than 100%.

For our evaluation we use the data gathered during 5 minutes before the alarm to generate the fingerprint. The operator sequence is generated for the duration of the alarm, i.e., for the period of time in which the alarm condition is true.

TABLE I. Dataset 1: The five most frequently raised alarms

Alarm number	Alarm count	With op-sequence	Different op-sequences
1101	171	57	12
1012	107	86	19
12	106	53	19
22	106	49	16
2	59	30	10

A. Dataset 1 – Simulation

As a first dataset we use a slightly extended version of the data we used in [1]. This dataset was obtained by operating a simulation model of the production machine. It contains $N_{Rows} = 1,150,063$ rows and $N_{Alarms} = 1,254$ alarms.

As in [1] we focus on alarm 1101. There are 171 instances of this alarm in the dataset. For 57 of these instances we could generate the 12 different operator sequences shown in Table II. The remaining 114 occurrences of the alarm do not contain operator sequences because the simulation was not always operated properly.

8 of these operator sequences corresponding to 10 occurrences of alarm 1101 are only contained in one of the four subsets. Thus, these sets are either part of the training data or of the test data. We therefore reduce the maximum possible score from 57 to 47.

For the event based algorithms we found 3 ambiguous event sequences. Table VII shows all event sequences. The ambiguous event sequences have more than one corresponding operator sequence. We consider ambiguous sequences as unlearnable. Our algorithm will always recommend the sequence with the highest frequency. Therefore we subtract the number of the other sequences from the maximum possible. In our case this reduces the maximum possible by 10 to 37.

For event sequence based algorithms we get a maximum possible score of

$$Q_{alg_{1ev_{max}}} = \frac{37}{57} \cdot 100\% \approx 65\% \tag{5}$$

For the point based algorithms we get a maximum score of

$$Q_{alg_{1p_{max}}} = \frac{47}{57} \cdot 100\% \approx 82\% \tag{6}$$

B. Dataset 2 – Converted Data from real Machine

The second set of process- and operator values was taken from a production machine.

The database of that machine does not yet conform to our interface standard and thus did not log the alarms. We resorted

TABLE II. Dataset 1: Operator sequences

Frequency	Operator Sequence	Learnable
36	`Loop1_3_SP_170`	yes
7	`Loop1_2_SP_170#Loop1_3_SP_170`	yes
4	`Loop1_4_SP_170`	yes
2	`Loop1_3_SP_250`	no
1	`Loop1_3_SP_225`	no
1	`AOUT1_1_OutValue_10`	no
1	`AOUT1_1_OutValue_29`	no
1	`AOUT1_1_OutValue_50`	no
1	`AOUT1_2_OutValue_9`	no
1	`Loop1_7_SP_250`	no
1	`Loop1_2_SP_250#Loop1_3_SP_250`	no
1	`Loop1_3_SP_170#Loop1_4_SP_170`	no

 ISBN: 978-1-61208-592-0

to generating our own alarms based on process experts' definitions for lower and upper bounds of process values. The first occurrence of a value being outside the bounds was the starting time for an alarm. The stopping time was taken the moment the value was back inside the bounds.

The resulting database contains $N_{Rows} = 1,223,992$ rows describing the machine state and $N_{Alarms} = 5,667$ alarms. Table III shows the five most frequent alarms in the dataset. With the machine logging the data every 10 seconds we have a runtime of $\approx 3,400$ hours. This means we have 1.7 alarms per hour. According to our project partner this number seems to be very high.

Not all alarms we generated from the machine were useful for our test. Some alarms were not followed by operator events. We assume that these alarms are artifacts of our data preparation algorithm. Other alarms obviously occured at the end of a production shift and consequently lead to the recommendation to shut down the machine. We chose alarm 225 for our evaluation.

Alarm 225 occurred 1949 times. For 277 instances of this alarm we could create a fingerprint and an operator sequence.

We assume that this difference is partly due to our self generated alarms and partly due to spurious and short alarms that disappear without user intervention.

For these 277 instances of alarm 225 we generated the 22 operator sequences shown in Table IV. 13 of these operator sequences corresponding to 46 event sequences occur in only one of the four subsets. Consequently we reduce the theoretical maximum by 46. Furthermore, we found 33 ambiguous event sequences. We subtract all but one of these ambiguous sequences except for those that are already marked as not learnable because they appear in only one subset. So we have to subtract further 47 alarms.

So for event based algorithms we get a maximum score of:

$$Q_{alg_{2ev_{max}}} = \frac{184}{277} \cdot 100\% \approx 66\% \quad (7)$$

For the point based algorithms we get a maximum score of:

$$Q_{alg_{2p_{max}}} = \frac{231}{277} \cdot 100\% \approx 83\% \quad (8)$$

C. Dataset 3 – Real Data

The third data set was gathered on a new production machine equipped with our system. The machine being new it is not yet fully productive and there is no long history of operation. We logged $N_{Rows} = 417,663$. As in the previous dataset a new row of data has been logged every 10 seconds. So we got a total run time of $\approx 1,200$ hours. During this time period we found $N_{Alarms} = 2,277$ alarms.

TABLE III. DATASET 2: THE FIVE MOST FREQUENTLY RAISED ALARMS

Alarm number	Alarm count	With op-sequence	Different op-equences
225	1949	277	22
423	1059	70	43
122	763	243	93
123	503	149	69
214	238	142	81
213	109	76	60

TABLE IV. DATASET 2: OPERATOR SEQUENCES FOR ALARM 225

Frequency	Operator Sequence	Learnable
54	AOut1_1_85#	yes
53	AOut1_1_84#	yes
30	AOut1_1_83#	yes
30	AOut1_1_86#	yes
20	AOut1_1_81#	yes
17	AOut1_1_82#	yes
16	AOut1_1_77#	no
14	AOut1_1_87#	yes
11	AOut1_1_80#	yes
9	AOut1_1_76#	no
7	AOut1_1_78#	no
5	AOut1_1_79#	no
2	AOut1_1_88#	yes
1	AOut1_1_0#AOut2_1_6#	no
1	AOut1_1_83#AOut2_1_64#	no
1	AOut1_1_83#AOut2_1_98#	no
1	AOut1_1_84#AOut2_1_98#	no
1	AOut1_1_85#AOut2_1_98#	no
1	AOut1_1_87#AOut2_1_73#	no
1	AOut1_1_89#	no
1	AOut1_1_91#Loop2_2_SP_230#	no
1	AOut2_1_99#	no

For this particular machine we can determine if the machine is in the production process or is idling. So we filtered for alarms being raised while the machine was running, we found $N_{Alarms} = 1,393$ alarms during operation. For 412 of these we could determine an operator sequence. Along with that we found 381 alarms which only occurred for a short time period, in this case less than 10 seconds. Table V shows the five most frequent alarms.

For further investigation we chose alarm 21.5 which occured 321 times. For 45 instances of the alarm we were able to create a fingerprint and an operator sequence. We used these alarms for our test. Table VI shows the operator sequences we could determine. In total we found 31 different operator sequences.

These operator sequences cover 17 alarms that we consider as learnable. The other ones occurring only once are not learnable by our algorithms. From the unique operator sequences 28 occurred in only one set. 28 alarms belong to these operator sequences. In this dataset we did not find any ambiguous event sequences. So for event and point based algorithms we get a maximum score of:

$$Q_{alg_{3ev_{max}}} = Q_{alg_{3p_{max}}} = \frac{17}{45} \cdot 100\% \approx 38\% \quad (9)$$

V. COMPARISON OF THE ALGORITHMS

After processing the three datasets through all of our algorithms we calculated the score for each dataset according to eq. 4. As discussed in sec. IV the resulting values for Q_{alg} are not comparable between the datasets because of unlearnable and ambiguous operator event sequences.

TABLE V. DATASET 3: THE FIVE MOST FREQUENTLY RAISED ALARMS

Alarm number	Alarm count	With op-sequence	Different op-equences
21.5	321	45	31
42.0	121	9	9
36.0	106	14	14
38.0	82	2	2
18.2	46	24	22
17.3	43	28	25

 ISBN: 978-1-61208-592-0

TABLE VI. DATASET 3: OPERATOR SEQUENCES FOR ALARM 21.5

Frequency	Operator Sequence	Learnable
13	ButtonDrive_1	yes
2	ButtonDrive_1# OutScrewspeed_30	yes
2	ButtonDrive_1# ToolButtonTemp_1	yes
1	ButtonDrive_0	no
1	ButtonDrive_0# MeltpumpOut_30	no
1	ButtonDrive_1# ButtonTemp_1# MeltpumpOut_20	no
1	ButtonDrive_1# ButtonTemp_1# MeltpumpOut_30# ToolButtonTemp_1	no
1	Button.Drive_1# OutScrewspeed_11	no
1	Button.Drive_1# OutScrewspeed_13# MeltpumpOut_10	no
1	Button.Drive_1# OutScrewspeed_14# MeltpumpOut_50	no
1	Button.Drive_1# OutScrewspeed_15# MeltpumpOut_13	no
1	Button.Drive_1# OutScrewspeed_19# MeltpumpOut_11# ToolTemp3SP_180	no
1	Button.Drive_1# OutScrewspeed_24# MeltpumpOut_37	no
1	Button.Drive_1# OutScrewspeed_27# ToolTemp1SP_195# ToolTemp2SP_185# ToolTemp3SP_180	no
1	Button.Drive_1# OutScrewspeed_28# Temp2SP_220#Temp4SP_220# ToolButtonTemp_1# ToolTemp1SP_195# ToolTemp3SP_198	no
1	Button.Drive_1# OutScrewspeed_43# MeltpumpOut_17	no
1	Button.Drive_1# OutScrewspeed_48	no
1	Button.Drive_1# OutScrewspeed_51	no
1	Button.Drive_1# OutScrewspeed_59# Meltpump.Out_40	no
1	Button.Drive_1# HaulOffButton_0	no
1	Button.Drive_1# MeltpumpOut_0	no
1	Button.Drive_1# MeltpumpOut_12	no
1	Button.Drive_1# MeltpumpOut_24# ToolButtonTemp_1	no
1	Button.Drive_1# MeltpumpOut_30	no
1	OutScrewspeed_15	no
1	OutScrewspeed_32	no
1	OutScrewspeed_34	no
1	Meltpump.Out_0	no
1	Meltpump.Out_14	no
1	Meltpump.Out_26	no
1	Meltpump.Out_48	no

Eqn. 5 to 9 give the maximum possible score for each dataset. With this score we can normalize the results as

$$Q_{alg_{rel}} = \frac{Q_{alg}}{Q_{alg_{max}}} \cdot 100\% \tag{10}$$

To put the performance into perspective we calculate the performance of a simple approach that always recommends the most frequent user sequence in the knowledge base. Such an approach would propose a correct operator sequence for all the alarms associated with the most frequent operator sequence:

$$Q_{simple} = \frac{100\%}{N_{OpSeq}} \cdot N_{mostFreqOpSeq} \tag{11}$$

i.e.,

$$Q_{simple_1} = \frac{100\%}{57} \cdot 36 \approx 63\% \tag{12}$$

$$Q_{simple_2} = \frac{100\%}{277} \cdot 54 \approx 19\% \tag{13}$$

$$Q_{simple_3} = \frac{100\%}{45} \cdot 13 \approx 28\% \tag{14}$$

Figure 3 shows the normalized performance of the different algorithms on the three datasets and for reference the normalized scores of the simple approach.

The mapping algorithm introduced in [1] and III-A1 scores $\approx 63\%$ on dataset 1. On dataset 2 it performed less with only $\approx 12\%$. On dataset 3 it did not give any correct recommendations. We attribute this lack of performance to the fact that this dataset does not have two identical event sequences. Since the algorithm performs a one to one matching it is not capable of finding any solution. In total it always scores less than the simple approach.

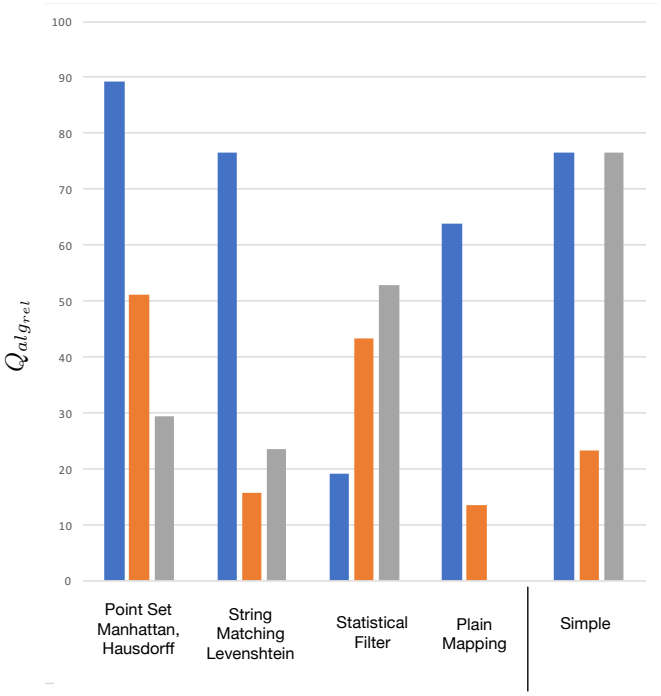


Figure 3. Normalized results for various recommender algorithms (left bars: Dataset 1, middle bars: Dataset 2, right bars: Dataset 3)

With $\approx 42\%$ on dataset 2 the statistical filter performs better than the simple approach and is almost on par with the point based algorithms. It also shows the best performance on dataset 3 but is still behind the simple algorithm.

The point based algorithm scores better on dataset 1 with almost 90% and $> 50\%$ on dataset 2. Dataset 3 however proves difficult with $\approx 29\%$.

VI. CONCLUSION AND FUTURE WORK

We used three datasets to generate and evaluate recommendations in "batch-mode". On some datasets our recommender algorithms outperformed a simple algorithm by a factor of 2. On our newest and smallest dataset, i.e., dataset 3, they lacked performance. That being said, we want to point out that all algorithms need the operator sequences we generate from the logged data.

Nevertheless, we will need to look further into dataset 3 and enlarge our knowledge base from $\approx 1,200$ hours and $\approx 2,000$ alarms to at least 5,000 hours and 10,000 alarms.

A quick analysis of dataset 3 showed that many alarms occur at the same time or in a very short timespan. So we will investigate whether it is easier to find a recommendation for groups of alarms.

We started interviewing experienced operators. We learned that in some situations the appropriate operator action is to just wait for the alarm to clear itself. So it might be a good idea to include a "do nothing" recommendation in the knowledge base.

 ISBN: 978-1-61208-592-0

We also plan to give the operator the possibility to evaluate our recommendation and to factor that evaluation into our recommendation.

REFERENCES

[1] V. Plenk, "Improving special purpose machine user-interfaces by machine-learning algorithms," in Proceedings of CENTRIC 2016 : The Ninth International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, Rome, August 2016, pp. 24 – 28.

[2] P. K. Wonga, J. Zhonga, Z. Yanga, and C. M. Vong, "Sparse bayesian extreme learning committee machine for engine simultaneous fault diagnosis," in Neurocomputing, 2016, pp. 331 – 343.

[3] K. Zidek, A. Hosovsky, and J. Dubjak, "Diagnostics of surface errors by embedded vision system and its classification by machine learning algorithms," in Key Engineering Materials, 2015, pp. 459 – 466.

[4] J. Luo, C.-M. Vong, and P.-K. Wong, "Sparse bayesian extreme learning machine for multi-classification," in IIEEE Transactions on Neural Networks and Learning Systems, 2014, pp. 836 – 843.

[5] A. Ziaja, I. Antoniadou, T. Barszcz, W. J. Staszewski, and K. Worden, "Fault detection in rolling element bearings using wavelet-based variance analysis and novelty detection," Journal of Vibration and Control, vol. 22, no. 2, February 2016, pp. 396–411.

[6] C. Challiol, A. Fortier, S. Gordillo, and G. Rossi, "A Flexible Architecture for Context-Aware Physical Hypermedia," in 18th International Workshop on: Database and Expert Systems Applications, 2007. DEXA '07., September 2007.

[7] S. Sigg, S. Haseloff, and K. David, "Context Prediction by Alignment Methods," in Proceedings of the 4th International Conference on Mobile Systems, Applications, and Services (MobiSys 2006), June 2006.

[8] S. Sigg, D. Gordon, G. von Zengen, M. Beigl, S. Haseloff, and K. David, "Investigation of Context Prediction Accuracy for Different Context Abstraction Levels," IEEE Transactions on Mobile Computing, vol. 11, no. 6, June 2012, pp. 1047 – 1059.

[9] V. Plenk, S. Lang, and F. Wogenstein, "Providing user guidance in special purpose machines by machine-learning algorithms," To appear in International Journal On Advances in Software, vol. 10, no. 3 and 4, December 2017.

[10] M. P. J. van der Loo, "The stringdist Package for Approximate String Matching," The R Journal, vol. Vol. 6/1, 2014, pp. 111 – 122.

[11] J. Lunze, Ereignisdiskrete Systeme. München: Oldenbourg, 2006.

TABLE VII. DATASET 1: EVENT SEQUENCES AND OPERATOR SEQUENCES

EventSequence	Operator Sequence	Frequency	Learnable
Empty[1]	Loop1_3_SP_225	1	no
	Loop1_3_SP_250	2	no
	AOUT1_1_OutValue_29	1	no
	Loop1_3_SP_170	2	yes
	Loop1_2_SP_170#Loop1_3_SP_170	1	yes
	Loop1_2_SP_250#Loop1_3_SP_250	1	no
	AOUT1_1_OutValue_10	1	no
Loop1_4_PV_B#Loop1_4_PV_X	Loop1_4_SP_170	1	yes
Loop1_3_SP_0.0	Loop1_3_SP_170	28	yes
	Loop1_2_SP_170#Loop1_3_SP_170	6	yes
Loop1_4_PV_H#Loop1_4_SP_50.0# Loop1_4_PV_G#Loop1_4_PV_F	Loop1_4_SP_170	1	yes
Loop1_4_SP_0.0	Loop1_4_SP_170	2	yes
Loop1_2_SP_0.0#Loop1_4_SP_0.0	Loop1_3_SP_170#Loop1_4_SP_170	1	no
	Loop1_3_SP_170	2	yes
Loop1_3_SP_10.0	Loop1_3_SP_170	1	yes
Loop1_4_PV_H#Loop1_5_PV_H# Loop1_6_PV_H#Loop1_7_SP_50.0	Loop1_7_SP_250	1	no
Loop1_3_SP_60.0	Loop1_3_SP_170	3	yes
AIN2_1_Value_G#MP1_Value_G#MT1_Value_F# MP1_Value_F#AIN2_1_Value_F#MP1_Value_B# MT1_Value_E#MP1_Value_C	AOUT1_2_OutValue_9	1	no
MP1_Value_E#MP1_BandMin_E#MP1_BandMax_E# Loop1_6_PV_E#Loop1_6_PV_F#Loop1_6_PV_E# Loop1_6_PV_D#Loop1_6_PV_C#Loop1_6_PV_B# Loop1_6_PV_A#MP1_BandMax_F#MP1_Value_H# MP1_BandMax_H#MP1_BandMin_F	AOUT1_1_OutValue_50	1	no

[1] In our simulation empty event sequences can occur when an alarm is generated by changing the alarm condition. This happens during testing or demonstration.

TABLE VIII. DATASET 2: AMBIGIOUS EVENT SEQUENCES

Event-Sequence	Operator-Sequence	Frequency	Learnable	Event-Sequence	Operator-Sequence	Frequency	Learnable
1	AOut1_1_84#	2	yes	18	AOut1_1_86#	2	yes
	AOut1_1_82#	1	yes		AOut1_1_87#	1	yes
2	AOut1_1_86#	3	yes	19	AOut1_1_85#	5	yes
	AOut1_1_87#	1	yes		AOut1_1_85#AOut2_1_98#	1	no
	AOut1_1_85#	1	yes		AOut1_1_84#	1	yes
3	AOut1_1_86#	2	yes	20	AOut1_1_82#	1	yes
	AOut1_1_87#	1	yes		AOut1_1_81#	1	yes
4	AOut1_1_84#	5	yes	21	AOut1_1_85#	1	yes
	AOut1_1_85#	4	yes		AOut1_1_87#	1	yes
5	AOut1_1_83#	2	yes	22	AOut1_1_87#	3	yes
	AOut1_1_85#	2	yes		AOut1_1_85#	1	yes
	AOut1_1_84#	1	yes		AOut1_1_86#	1	yes
6	AOut1_1_80#	2	yes	23	AOut1_1_83#	1	yes
	AOut1_1_79#	1	no		AOut1_1_81#	1	yes
7	AOut1_1_87#	1	yes	24	AOut1_1_81#	2	yes
	AOut1_1_86#	1	yes		AOut1_1_80#	1	yes
8	AOut1_1_85#	1	yes	25	AOut1_1_85#	1	yes
	AOut1_1_84#	1	yes		AOut1_1_86#	1	yes
9	AOut1_1_83#	1	yes	26	AOut1_1_80#	3	yes
	AOut1_1_84#	1	yes		AOut1_1_81#	1	yes
10	AOut1_1_84#	2	yes	27	AOut1_1_84#	1	yes
	AOut1_1_83#	1	yes		AOut1_1_85#	1	yes
11	AOut1_1_82#	1	yes	28	AOut1_1_77#	3	no
	AOut1_1_83#	2	yes		AOut1_1_78#	1	no
12	AOut1_1_86#	2	yes	29	AOut1_1_85#	1	yes
	AOut1_1_88#	1	yes		AOut1_1_86#	1	yes
	AOut1_1_87#	1	yes		AOut1_1_84#	2	yes
13	AOut1_1_87#	1	yes	30	AOut1_1_82#	1	yes
	AOut1_1_86#	1	yes		AOut1_1_81#	1	yes
14	AOut1_1_86#	2	yes	31	AOut1_1_85#	1	yes
	AOut1_1_87#	1	yes		AOut1_1_86#	1	yes
15	AOut1_1_85#	2	yes	32	AOut1_1_83#	1	yes
	AOut1_1_86#	2	yes		AOut1_1_85#	1	yes
16	AOut1_1_84#	1	yes	33	AOut1_1_84#	1	yes
	AOut1_1_83#	1	yes		AOut1_1_83#	1	yes
	AOut1_1_85#	1	yes		AOut1_1_85#	2	yes
	AOut1_1_82#	1	yes				
17	AOut1_1_83#	1	yes				
	AOut1_1_85#	7	yes				
	AOut1_1_84#	3	yes				

 ISBN: 978-1-61208-592-0