Service Oriented Machine-learning Application for Reconfigurable Predictive Maintenance System

M. Mostafizur Rahman, Nandini Chakravorti, Nils Weinert, Robert Munnoch, Swati Jadhav

ABSTRACT—Successfully deploying AI models for a reconfigurable AI-driven enterprise is a challenge. This work focuses on the design and implementation of a reconfigurable machine learning data analytics tool supporting flexible and automatic configurability in manufacturing. This tool provides a user interface and RESTful API to be used by other components of the architecture for deployment and use of machine learning and statistical models as a service for the purpose of predictive maintenance. This tool supports text classification to label unstructured data and uses machine-learning models based on decision-trees to predict failure of the production machines.

Index Terms —machine learning, condition monitoring, predictive maintenance, text mining, service orientated architecture

I. Introduction

Manufacturing industry is facing a myriad issues regarding the availability and reliability of the equipment. Equipment faults can cause long production line stoppages, high maintenance costs and low product quality. Well planned maintenance assists with keeping the equipment in a healthy condition, and helps reduce the risk of large scale machine damages and revenue loss. There are mainly three categories of maintenance methodologies: reactive, preventive and predictive (condition-based) maintenance methods [1].

Reactive maintenance focuses on repairing an asset once the failure occurs. The main problems associated with the reactive maintenance approach include: unpredictability of run time, costs associated with downtime, high cost of maintenance and risk of catastrophic failures.

Preventive maintenance focuses on avoiding failures by performing maintenance tasks at predetermined intervals depending on the operating conditions. The main issues associated with preventive maintenance include over-maintenance of the machine, high cost of changing and storing unnecessary parts and expensive skilled personnel.

Manuscript received 14th March, 2019; revised 29th March, 2019. This work was supported by European Union's Horizon 2020 research and innovation programme under grant agreement No 68043.

M M Rahman is a Technical Specilist for Data and Information System at The Manufacturing Technology Centre, one of the High Value Manufacturing CATAPULT in UK, e-mail: Mostafizur.rahman@the-mtc.org .

N Chakravorti is the Technology Manager for the Data and Information System at The Manufacturing Technology Centre, Coventry, UK, e-mail: Nandini.Chakravorti@the-mtc.org .

R A Munnoch is a Senior Research Engineer for the Data and Information System at The Manufacturing Technology Centre, Coventry, UK, Robert.Munnoch@the-mtc.org .

S Jadhav is an Advance Research Engineer for the Data and Information System at The Manufacturing Technology Centre, Coventry, UK, e-mail: Swati.Jadhav@the-mtc.org .

Nils Weinert is from Siemens AG's Corporate Technology, Technology Field Simulation and Digital Twin, Munich, Germany, e-mail: Nils.Weinert@siemens.com .

Predictive maintenance relies on condition monitoring techniques to predict the occurrence of a failure by monitoring relevant sensor data during the machine operation. This allows the maintenances to be planned and even prevented if appropriate. This also lends to easier and more reliable maintenance plans since run time and availability can be relied upon. Maintenance only happens when parts are actually wearing out and can aid leaner operations with provable cost benefit analysis.

Condition based predictive maintenance is considered to be the best approach for improving equipment reliability, reducing production costs due to failure, reducing costs due to maintenance, optimising maintenance intervals, reducing the risks of catastrophic damage of the health of the machines and minimising unplanned downtime. However, in real production environment not all the failures and downtime reported are due to the condition of equipment. Many of the failures and downtime are caused by the production process itself rather than the condition of the machines. Hence, proper insight on the problem by analysing historical data is very important for selecting an appropriate maintenance task.

The objective of this paper is to present a Service-Oriented Machine Learning framework for a reconfigurable predictive maintenance system. The results of the application of the framework on the production of industrial compressors and gas separators use case is also detailed. The work done within the paper is a part of the H2020 project, Production harmonized Reconfiguration of Flexible Robots and Machinery (PERFoRM) which is focussed on the requirements of increasing flexibility and configurability in manufacturing.

II. Use case

The factory selected as an use case is responsible for the manufacturing of tailored compressors trains for oil and gas applications, such as air separation units or for the Liquefied Natural Gas (LNG) production. Currently the production of compressors is characterised by small lot sizes, machining, manual labour and a highly complex final assembly.

The objective of this use case is to improve the flexibility of manufacturing, in particular focusing on the mechanical manufacturing of housing parts. The intention is to use predictive and condition based maintenance, which shall improve the flexibility of manufacturing by better shifting production tasks between different machines and reduce quality issues and production failures. A failure or machine breakdown can lead to delays of the productions and missing parts to semi-finished products. The deployment of a predictive maintenance system involves the monitoring of the health of the equipment (three turning machines within the current context). The tool will support the maintenance

ISBN: 978-988-14048-6-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

engineer to generate related maintenance tasks before the machine actually fails. These tasks can then be combined with production tasks in the overall production schedule.

III. Service Oriented Machine-Learning Framework

The overall architecture of the use case can be seen in Figure 1. The architecture includes a Maintenance ticketing system and data (referred to as PDA/MDA in Figure 1) which includes the production data from the ERP (Enterprise Resource Planning) system and machine alarms data. It is envisaged that the machine conditions will be monitored by additional sensors which will automatically publish the data to the middleware shown in Figure 1. Technology adaptors, used for seamless exchange of data within heterogenous devices will be used to acquire live data from the databases involved. Additionally, standard interfaces (letter "S" as seen in Figure 1) will be used to enable plug-ability and interoperability. In order to enable each component within the architecture to recognise and communicate with each other, a middleware (referred to as the PERFoRM middleware in Figure 1) has been introduced. Further details can be found in our previous work [2].

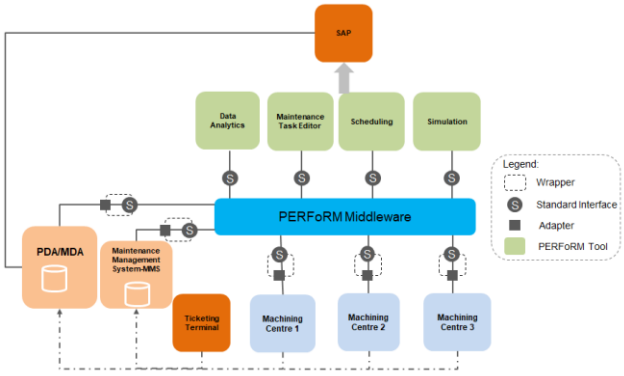

Figure 1 Siemens overall architecture

The Data Analytics tool will access the data via the middleware, analyse it and visualise the result. The results generated by the Data Analytics tool will be used for the manual creation of new maintenance tasks within the Maintenance Task Editor. The Scheduling tool then accesses the maintenance tasks and proposes schedules for production and maintenance tasks. After evaluation of the schedules are done by the Simulation tool, the most appropriate maintenance task will be transferred to the ERP system. It is to be noted that this paper only presents the functionalities of the Data Analytics tool and its interfaces.

A. *Proposed framework for data analysis and visualisation*

A framework has been developed to allow user to analyse the data, process it and get different results (see Figure 2). The data available to the framework are: (1) machine alarms and (2) information from the Maintenance ticketing system. The maintenance ticket contains description of the failures; it is mainly entries representing the date when a failure occurred and a textual representation giving details of the failure and sometimes what action has been taken. The production data acquisition system contains several types of data (PDA/MDA), however, this work uses the alarms data part of this database. This data represent entries with their timestamp and alarm numbers that has been triggered by the machines. Each row represents one triggered alarm at a particular moment. The proposed framework also utilises unstructured data from the appropriate maintenance manuals.

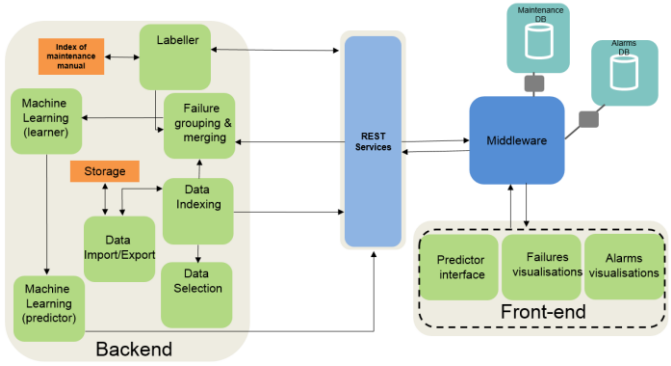

Figure 2 Internal architecture for Data Analytics service

As seen in Figure 2, the data will be transferred from the databases over the PERFoRM middleware. It includes a Backend System for processing the data where the different algorithms are implemented. .The Front-end is used for visualising the failure data and the alarms. The frontend collects the pre-processed data from the backend and produces plots for visualising failures and key alarms.A REST interface has been included to export the functionalities of the Backend System within the proposed architecture

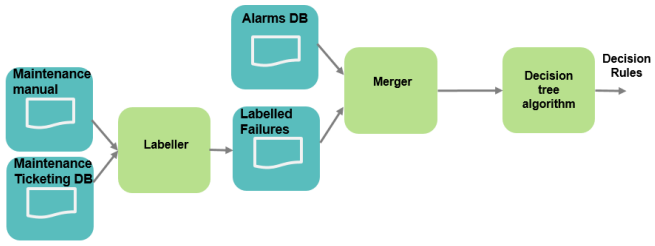

Figure 3 Machine Learning components within the architecture

The Backend system mainly uses the alarms data, maintenance data and the machine maintenance manual. The maintenance database contains description of the problem reported in an unstructured textual form. This textual form is not the actual failure type, however it may be possible to infer the failure type manually from the text. Consequently, prior to any automated processing of this data, it needs to be appropriately and consistently labelled. Within the current work, the maintenance manual is being used as an input for automated labelling of the textual problem description (further details can be found in Section B). Figure 3 depicts the sequence of operations and the components involved in producing decision rules to predict failures The different components of the system are as follows:

Failure grouping/merging: This component labels a failure with an textual description. Labelling involves production of an automatically generated type of the failure for each description (from maintenance database). Section B gives more details about the underlying structure of the Labeller

ISBN: 978-988-14048-6-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

module. This step is necessary since identifying the type of failures can play an important role in predicting them.

Data indexing and Data Selection: Once the data is acquired through the middleware, an indexer will process it using the dates and their identifier (alarm number or failure type). The purpose of this step is to allow a faster query response for a range of dates/types/number. Consequently, the selection of the relevant data (depending on the visualisations) will be done in few memory accesses. One of the requirements of the visualisation is to group the data by range of dates which is variable and sub-selections based on alarm number of failure types. The dataset represents more than one million of entries. This number is estimated to go higher over time. With these settings, an indexing mechanism is a necessity. For this, a combination of data structures have been used: the hash maps and the red-black trees. Hash maps can be very effective when querying a single element (an alarm or a failure) based on a key while red-black trees can be the best choice when dealing with a range of keys. For the hash maps, the keys are the alarm number or the failure type (depending on which dataset). For the red-black tree the keys are the dates. Hash maps consisted of keys of alarm number or failure types while their entries were red-black trees. In other words, for each alarm number or failure type there is a red-black tree [3]. The red-black tree contained the indexed alarms or failures by date, e.g. if a visualisation requires the alarms that happened between January 2016 and July 2016 for the alarm number 1000. First, the red-black tree required is brought from the hash map using the alarm number 1000. Once the required red-black tree is fetched, a sub-selection based on dates is applied to the red-black tree. To access the red-black tree from the hash map only one memory access is required (in most cases) while access elements in a red-black tree requires log(n) accesses where n is the number of elements in the tree. Results of this indexing strategies showed a considerable improvement over queries using non indexed data. On average, queries using indexing took 16ms to return the results back whereas they took >100ms in the case of non-indexed data (using a regular core i5-4200m CPU).

Data export/import module: A export/import module is used to store the indexed data on disk. The loader module will load the saved data from the disk into memory, thus avoiding loading the data again from the middleware in case of a failure. In the case of an update, only the new data will be loaded from the middleware and then dumped to the local disk.

Data Merging and Balancing: Alarm data and failure data are two separate entities. To be able to create a link between the two entities and infer some rules, a dataset merging alarms and failures needs to be produced. Each row is composed of the presence or not of the list of alarms and the class of a failure in it has been observed. Thus each row will represent an association between the set of alarms and the presence of a failure or not: $row_i=(a_1,a_2,\ldots,a_n,F)$, where n is the number of alarms, F the type of present failure and i represents a row that corresponds to a particular day. This association is necessary for creating a machine learning model that can predict failures based on the alarms observed on a day i. After merging the data, the problem of class imbalance was noticed i.e., the row constituting the presence of failures were considerably less than those where no failures were observed (the number of days where no failures happened was far greater than the number of days where a failure happened). This problem can particularly impact the accuracy of any machine learning model for failures prediction. A technique based on data augmentation has been used to address the problem of class imbalance and is described in section IV-A.

Machine learning for predictive model: After pre-processing (failure labelling and combining the alarms and failure data) the data and balancing the data, machine learning techniques are applied to generate a predictive model to extract decision rules. The Decision tree classifier is one of the most widely used machine learning methods. Decision tree models are commonly used in data mining to explore and classify the data. The induced tree and its associated rules will be used to make predictions [4]. Details of the data mining and analytics approach are explained in section IV-C.

Web services / REST: The goal of the Data Analytics tool is to support re-configurability and extensibility of the system in a service orientated architecture. To achieve this, several of the capabilities of this tool were made accessable using REST services. This allows them to be exposed through the middleware to decouple the backend services from the frontend interface. This allows a richer frontend user experience that can utilise multiple back-ends for its visualisations. One other advantage of using web services is that they can be documentated and described as an API to be used by other tools/services within the architecture.

IV. MACHINE LEARNING METHODS USED

A. Data Pre-processing using SMOTE over-sampling

A balanced dataset is very important for creating a good training set. Most existing classification methods tend to perform poorly on minority class examples when the dataset is extremely imbalanced. These methods aim to optimise the overall accuracy without considering the relative distribution of each class [5].

The data used in the current case study was observed to be highly imbalanced, with very few records for each failure. Good sampling strategies are required to overcome this problem. In the current context, an over-sampling technique called SMOTE [6] has been used wherein the minority class is over-sampled by creating "Synthetic" examples. This ensures that the dataset represents all types of failures fairly, thus allowing the produced model to be less biased.

B. Failures Labelling

Unstructured data can pose a challenge in predictions. The meaningful features that allow effective predictions can take different forms across the dataset. This challenge was observed in the maintenance tickets dataset (failures dataset). This dataset consists of operator entries in plain text. Other entries in the dataset include: timestamp, name of the human operator who recorded the failure and its textual description. The operator rarely records deterministic details about the actual failure such as what part of the machine failed or any other classification. This kind of textual representation pose a serious problem for automatic processing.

Knowing the class to which a particular failure belongs can help linking classes of failures with symptoms. Symptoms can include alarms data, sensors data and operations data. Thus, an

ISBN: 978-988-14048-6-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

appropriate machine learning model will be able to identify the symptoms related to certain classes of failures.

Under the current circumstances, building such a link is not possible. For this, a technique labelling the different textual description of failures and classifying them has been put together.

Various supervised machine learning models [7], [8], [9] can be used to classify data with a number of features. However, these supervised models require a training set of data where labels or classes are known in advance. Using this labelled data that includes features and to what class they belong to, they can learn to map these features with their respective labels/classes. Yet, using a pure supervised model will not be possible with the present dataset since a sample where failures are already labelled does not exist and all of the recorded failures are unlabelled.

Another strategy is to use clustering algorithms [10], [11]. These class of algorithms try to regroup similar data points in clusters. Each cluster consists of a number of data points belonging to the same label/class. In our case each cluster can represent a failure type and in each cluster a number of failures will be grouped sharing the label defined by that cluster. A problem with this method is that an expert intervention is required to be able to provide labels to clusters. A second disadvantage is that many failures that do not present the same similarities on the used keywords will be outliers and will not belong to any class. This is particularly true with human operator based descriptions where the used keywords can be quite irregular.

The method used in this work for labelling the textual description of failures is based on information retrieval [12], [13]. The idea is to use the maintenance manual of the machine as a labelling guide. This is done by looking at the keywords describing a failures and keywords used in certain sections of the maintenance manual. In the maintenance manual, each section deals with one component of the machine. There are 16 components in total. This has suggested the possibility of using 16 different labels for failures description. Another label "other" was added to deal with miscellaneous failures that may not be related to components directly, making 17 possible labels in total. The intuition behind this method is that keywords describing a particular failure have a high chance to appear in the maintenance manual's section dealing with the failed component and by consequence its label.

a) Solution formulation:

The primary material used as a guide to label the failures descriptions is the maintenance manual. This material was divided into 16 parts. Each part represents one section dealing with one (and only one) of the components of the machine. The text of these sections was used as a reference for labelling. The method used measures the distance between failure descriptions and the 16 sections of the maintenance manual. The closest section of the maintenance manual to a particular failure will become its label. The following pseudocode illustrates the procedure of labelling:

```
Function labelFailure(failureDescription): FailureLabel
    bestSimilarity=0
    rightSection
     For each section in maintenance manual:
     similarity=computeSimilarity(failureDescription,
section.getContentAsText())
     If similarity > bestSimilarity :
        bestSimilarity=similarity
        rightSection=section

return rightSection
```

b) Similarity computation

To compute the similarity between a failure description and one section of the manual, a cosine measure [14],[15] has been used:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \qquad \text{Equation 1}$$

The cosine similarity measure takes as inputs two vectors and produces a number describing the similarity between the two vectors. One vector represents the failure description while the other vector represents the maintenance manual section. The higher the cosine value the closer are the failure description and the manual's section. What needs to be described is how failures and manual's sections are converted into vectors. This text-to-vector conversion uses TF*IDF (term frequency * inverse document frequency) [15], [16]. The vector produced will contain the TF*IDF value for each existing term in the maintenance manual and the failure description. This means that the length of the vector is the total number of distinct words present in the collection. TF is the term frequency (the number of appearances of the term in failure description or the manual's section). IDF is the inverse document frequency which is, if a word is present in one section and absent in all other sections, then this word has higher chances to represent this particular section and then a higher IDF value is generated for that term. TF*IDF is simply the product of these two terms.

After converting the manual's sections into vectors, only these vectors will be stored on the system as they are sufficient to compute the similarity and the whole text of the maintenance manual is no longer required.

c) Compound words problem

The dataset that was used for this work is mainly in German (both the textual description and the maintenance manual). Germanic languages have a particularity of writing very long words which are combinations of other words. The absence of spaces in these words can make it particularly challenging for similarity measures, as related keywords might not be detected as similar ones but different ones due to their different stems. Search engines for Germanic languages have word decompounders included which has improved the results remarkably [17]. Similarly, a German decompounder has been written to add spaces in long words. The German decompounder is a dictionary based one. Words in a dictionary are scanned and compared if they exist within keywords of a certain length and then decompounded into many smaller words. This decompounder has been used only on the failures descriptions and not on the sections of the maintenance

ISBN: 978-988-14048-6-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

manual. The reason for that is that the maintenance manual has been written in a more formal style and as the maintenance manual is considerably larger than failure descriptions, decompounding its keywords added several noisy small words that reduced the accuracy.

d) Ontology

Another problem that was observed is that operators that wrote failures descriptions could use non normalised keywords, synonyms, descriptions that are not present on the maintenance manual or are written in a different style. To tackle this problem an ontology has been used that links one keyword to several other keywords or a group of keywords to one keywords. Keywords that are related are added to the failure descriptions. Example: if failure description contains the keyword A and on the ontology A and B are related, then the word B is added to the failure description. Thus making it more conform to the style of writing of the maintenance manual. Results are as follows:

Evaluation was conducted before the development of the decompounder and the ontology (see Table 1). As a consequence, these two modules' effectiveness has not been assessed directly. The procedure involved a human expert selecting randomly 30 distinct failures including their textual descriptions. The human expert labelled the failures using his domain knowledge. A comparison of the labels produced by the tool and by the expert was done. Results indicated that the accuracy of the tool ranged between 50%-60% despite the high number of classes (17 in total). An initial evaluation using both the decompounder and the ontology has indicated an accuracy level above 70%. However, the results produced by the previous evaluation may have influenced how the ontology was constructed. Thus an independent new evaluation needs to be conducted. Table 1 represents the results evaluation.

TABLE 1

SAMPLE OF THE RESULTS OF THE EVALUATION OF THE LABELLER

Störmeldung (failure description)	Label (predicted)	Evaluation
Hebel an der Bedienertür defekt	HYDRAULIK- UND SCHMIERMITTEL KREISLÄUFE	No
Späneförderer (Sicherung)	SPÄNEFÖRDERER	Yes
Pumpenmotoren Hydraulikaggregat überprüfen.Hydraulikbehälter kontrollieren.	TISCH	No
Maschine im Notaus	TISCH	Unlikely
Handrad funktioniert nicht.	ANDERE	Possible
Fehler bei Werkzeuggreifer	MASCHINENBETT	No
Monteurunterstützung	ANDERE	Principle
Schaltschranktür offen Klimaanlage läuft nicht. Trittbrett-Fehler, geht nicht wider zu nach dem ich es runter fahre.	ELEKTRISCHE AUSRÜSTUNG	Yes
Die Schutztür des Scheibenmagazins lässt sich nicht schließen. Fehlermeldung: 700702	SCHUTZKASTEN	Yes

TABLE 2

LIST OF RULES WITH THEIR ACCURACIES USING DECISSION TREE

Rule (alarm number)	Failure Decision	Accuracy
Other combinations (default)	No	N/A
10860 = No AND 20050 = No AND 700333 = No AND 510308 = Yes AND 700540 = No	Yes	95.65%
10860 = No AND 20050 = No AND 700333 = No AND 10621 = No AND 700310 = Yes	Yes	83.02%
10860 = No AND 20050 = No AND 700333 = No AND 700543 = No AND 10208 = Yes AND 67051 = No AND 700754 = No	Yes	80.43%
10860 = No AND 20050 = No AND 700333 = No AND 700543 = No AND 601012 = Yes AND 700636 = No	Yes	77.83%
10860 = No AND 20050 = No AND 700333 = No AND 700543 = No AND 700310 = Yes AND 700733 = Yes AND 16913 = Yes AND 67051 = No	Yes	77.36%
10860 = No AND 20050 = No AND 700333 = No AND 10621 = No AND 700310 = No AND 4075 = No AND 510216 = No AND 700239 = No AND 600609 = Yes AND 700635 = Yes AND 600908 = Yes AND 600410 = Yes AND 16906 = Yes AND 67050 = No	Yes	75.00%
10860 = No AND 20050 = No AND 510216 = No AND 700239 = No AND 600609 = Yes AND 700635 = Yes AND 600908 = Yes AND 600410 = Yes AND 700646 = Yes AND 510309 = Yes AND 10208 = Yes AND 6406 = No AND 700732 = Yes AND 601012 = Yes	Yes	69.39%

C. *Machine Learning for Predictive Models via Decision Tree*

Decision trees are a type of classifier that can produce rules from data. The advantage of using decision trees is that they are one of the few classifiers whose output is human readable and understandable. After merging the data, decision trees were employed to extract rules that predict if a failure is about to happen. For each type of failures several rules are produced.

Table 2 presents exemplary rules for one of the most common type of failures. It also includes the prediction accuracy for each rule. These rules are used by the monitoring system to trigger the state of the machines. The evaluation of the rules was conducted with a different dataset than the one used for training. 75% of the original data has been used for training to construct the rules while the rest 25% were used for evaluation.

D. *Validation of the Framework : Web services / REST*

The data analytics tool presents a number of functionalities such as automatic labelling of failures, fast data selection and data aggregation, and failure prediction. These capabilities are developed as a library which is used along with a web interface and also wrapped around REST services. REST services allowed to expose the data analytics functionalities to the other modules in the PERFoRM architecture, thus allowing services composition and reusability. Figure 4 shows a visualisation of aggregated alarms along with the observed failures.

ISBN: 978-988-14048-6-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

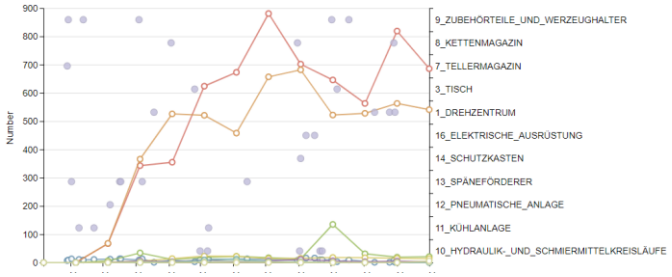

Figure 4: Alarm trends for failure groups per time period; failure groups represented on the right hand-side Y-axis. Dot represent the failures. The line show the aggregration on key alarms selected by the machine learning method over period of time.

V. Conclusion

A service oriented data analytics tool has been developed within the current work. This tool provides a user interface with a set of REST service API's to allow a higher level of reusability and modular integration. One of the novelties of this tool is to consolidate the meaning of the maintenance ticket's text and then link these to the alarms dataset. The first challenge encountrered regarding this functionality was that the maintenance ticket dataset had an unstructured format and requires a non conventional data processing method to be exploited. For this, a method based on text processing and information retrieval has been used, linking two unstructured sources which are the failures descriptions and the maintenance manual, making a structured dataset and labelling them deterministically. A word decompounder has been used to split long words in germanic languages. Then, the words were normalised using an ontology. The initial evaluation showed that linking the maintenance manual to the failure description gave favourable results (50%-60%) although the decompounder and the ontology were not used in this evaluation due to not being ready when the evaluation was carried out. Future work includes adding these into the evaluation. Data mining techniques have been employed to link the the alarm data to the failure. This enabled a prediction of failures (expressed as rules) based on the observed alarms. Some of the extracted rules showed a high level of accuracy (75%-95%). However, the rules extracted do not cover all the failures and finding good links between alarms and failures datahas proved to be challenging due to the low correlation between the two datasets. In the future, taking the sequence of alarms in the generated rules will be exploited. Different data processing algorithms have been implemented to help real time visualisations, such as quick selecting, sorting and grouping the data by time. The indexing stratergy used allowed operations to be done in real time. Finally, all of these functionalities are exposed through a RESTful service compliant with the PERFoRM architecture to allow the required integration. A web based user interface has also been developed. A full evaluation of the PERFORM architecture including this service is to be carried out in the future.

Acknowledgment

The PERFoRM project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 68043. Authors of this paper would also like to thank Dr. Mohamed R. Sidoumou for his support while he was working for the Manufacturing Technology Centre UK.

References

[1] M. Tomlinson, "What is the ideal maintenance strategy? A look at both MoD and commercial shipping best practice," *bmtdsl.co.uk*, 2016.
[2] N. Chakravorti, E. Dimanidou, G. Angione, J. Wermann, and F. Gosewehr, "Validation of PERFoRM reference architecture demonstrating an automatic robot reconfiguration application," in *IEEE 15th International Conference on Industrial Informatics (INDIN), Emden*, 2017, pp. 1167–1172.
[3] R. Bayer, "Symmetric binary B-Trees: Data structure and maintenance algorithms," *Acta Inform.*, vol. 1, no. 4, pp. 290–306, 1972.
[4] R. C. Barros, P. A. Jaskowiak, R. Cerri, and A. C. de Carvalho, "A framework for bottom-up induction of oblique decision trees," *Neurocomputing*, vol. 135, pp. 3–12, 2014.
[5] M. M. Rahman and D. N. Davis, "Addressing the class imbalance problem in medical datasets," *Int. J. Mach. Learn. Comput.*, vol. 3, no. 2, p. 224, 2013.
[6] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.
[7] T. Joachims, "Transductive inference for text classification using support vector machines," in *ICML*, 1999, vol. 99, pp. 200–209.
[8] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas, "Short text classification in twitter to improve information filtering," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 2010, pp. 841–842.
[9] S.-B. Kim, K.-S. Han, H.-C. Rim, and S. H. Myaeng, "Some effective techniques for naive bayes text classification," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 11, pp. 1457–1466, 2006.
[10] C. C. Aggarwal and C. Zhai, "A Survey of Text Classification Algorithms," in *Mining Text Data*, Boston, MA: Springer US, 2012, pp. 163–222.
[11] L. Liu, J. Kang, J. Yu, … Z. W.-. I. N.-K. P. of, and U. 2005, "A comparative study on unsupervised feature selection methods for text clustering," in *Natural Language Processing and Knowledge Engineering, 2005. IEEE NLP-KE'05. Proceedings of 2005 IEEE International Conference on, IEEE 2005*, 2005, pp. 597–601.
[12] H. Schütze, C. Manning, and P. Raghavan, *Introduction to information retrieval*. 2008.
[13] M. W. Berry, "Survey of Text Mining: Clustering, Classification, and Retrieval Springer-Verlag New York," *Inc.-2004.-244 p*, 2004.
[14] A. Bhattacharyya, "On a measure of divergence between two multinomial populations," *Sankhyā indian J. Stat.*, pp. 401–406, 1946.
[15] G. Salton and M. J. McGill, *Introduction to modern information retrieval*. McGraw-Hill, 1983.
[16] W. Frakes and R. Baeza-Yates, *Information retrieval: Data structures & algorithms*. prentice Hall, 1992.
[17] M. Braschler and B. Ripplinger, "How Effective is Stemming and Decompounding for German Text Retrieval?," *Inf. Retr. Boston.*, vol. 7, no. 3/4, pp. 291–316, Sep. 2004.