

# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900

Open access books available

186,000

International authors and editors

200M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index  
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?  
Contact [book.department@intechopen.com](mailto:book.department@intechopen.com)

Numbers displayed above are based on latest data collected.  
For more information visit [www.intechopen.com](http://www.intechopen.com)



# Text Mining for Industrial Machine Predictive Maintenance with Multiple Data Sources

*Giancarlo Nota and Alberto Postiglione*

## Abstract

This paper presents an innovative methodology, from which an efficient system prototype is derived, for the algorithmic prediction of malfunctions of a generic industrial machine tool. It integrates physical devices and machinery with Text Mining technologies and allows the identification of anomalous behaviors, even of minimal entity, rarely perceived by other strategies in a machine tool. The system works without waiting for the end of the shift or the planned stop of the machine. Operationally, the system analyzes the log messages emitted by multiple data sources associated with a machine tool (such as different types of sensors and log files produced by part programs running on CNC or PLC) and deduces whether they can be inferred from them future machine malfunctions. In a preliminary offline phase, the system associates an alert level with each message and stores it in a data structure. At runtime, three algorithms guide the system: pre-processing, matching and analysis: Preprocessing, performed only once, builds the data structure; Matching, in which the system issues the alert level associated with the message; Analysis, which identifies possible future criticalities. It can also analyze an entire historical series of stored messages. The algorithms have a linear execution time and are independent of the size of the data structure, which does not need to be sorted and therefore can be updated without any computational effort.

**Keywords:** industrial machine tool, predictive maintenance, log message, text mining, efficient algorithm

## 1. Introduction

The concept of Predictive Maintenance [1–4] foresees the carrying out of maintenance activities before the equipment failure. The primary goal of predictive maintenance is to reduce the frequency of equipment failures by preventing the failure before it actually occurs [5]. This strategy helps to minimize breakdown costs and downtime (loss of production) and increase product quality, well known thanks to [6] and recently reiterated by [4]. Obviously [7] predictive maintenance is different from corrective maintenance, as action will be taken here to “anticipate” the error before it actually occurs. Predictive maintenance is primarily about detecting hidden and potential failures. It does not replace, but joins the Preventive Maintenance in the strict sense, which is linked to the execution of a specific protocol (often agreed with the machine manufacturer) intended to periodically

check, or after a certain amount of work, the state of use of the machine, without any signs of behavioral anomalies having actually occurred. According to [8], if the maintenance strategy only involves interventions that react to failures, the maintenance costs are relatively low but the losses could be high. If preventive and predictive maintenance is introduced, maintenance costs increase: for example, some activities must be carried out using overtime, detectors for predictive maintenance are introduced, time is dedicated to training activities for operators and maintenance workers. Some clues for defining the algorithms on which the cyber-physical system presented here is based derive from research in the field of computational linguistics, which have appeared on papers presented at various important international conferences [9–14].

Here we present a discrete dynamic system, based on events represented as textual messages to which an alert level has been associated and whose data structure is a graph. The system is dynamic because the data structure adjusts itself without additional computational costs if a new message is issued by the machine, which has never appeared before and that is not yet included in the set of known messages. The new message must in any case be validated by a human expert, who must associate the message with an adequate alert level.

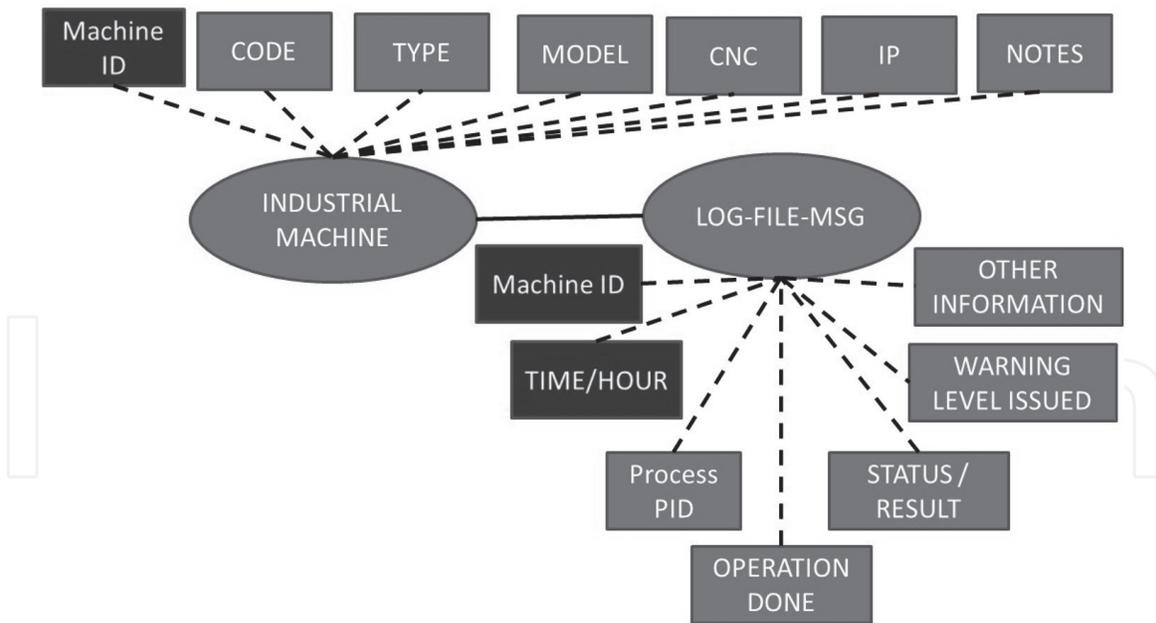
This paper is structured as follows: in section 2 we present the main characteristics of the data emitted by the data sources associated with an industrial machine tool, and how they are represented in the system. In section 3 we present the system. In particular, in section 3.1 we present the model for a machine tool, in section 3.2 we give an overview of the system, while in section 3.3 and in section 3.4 we present the two main phases of the system: pre-processing, to be performed only once, and runtime; in section 3.5 and in section 3.6 we present the main algorithms and their theoretical performances; in section 4 we report a brief summary of the results of a prototype of the system applied to a simple, but real, case study. The paper ends in section 5 with some conclusions.

## **2. Industrial machine tool data**

In the following, the term “machine data” defines an information message emitted by any “data source” associated with a machine tool (for example a sensor), which concerns an event that occurred during the activity of the machine itself. Machine data are the log files emitted by the machine itself, but also all data emitted by external entities such as event sensors, sensors for speed, temperature, acceleration and so on.

In order to reconstruct in detail the “history” of the messages, machine tool data must be raw and immutable (as opposed to classical structured and aggregated Relational Databases data). Data is never deleted or updated (except in very rare cases, for example to comply with regulations), it is only added. The main disadvantage of this management is that the stored data tends to become very large. From this point of view we can talk about big data. To avoid misinterpretation of the data, they are not stored in a free format, but are “semantically normalized”, that is, remodeled in a standard format, even if not strictly structured.

Machine tools data are represented through the “fact-based model” (see **Figure 1**): a graph where each message corresponds to a single fact. In this graph, each node corresponds to a machine tool data source entity (e.g. Sensor Y on machine M) and each arc can represent information about an entity (dashed connecting line) or a relationship between two entities (continuous line). Each single message is identified by its timestamp plus the identification of the machine (M) and the identification of the entity that emits the message (Sensor Y).



**Figure 1.**  
 A simple fact-based model for a standard log file.

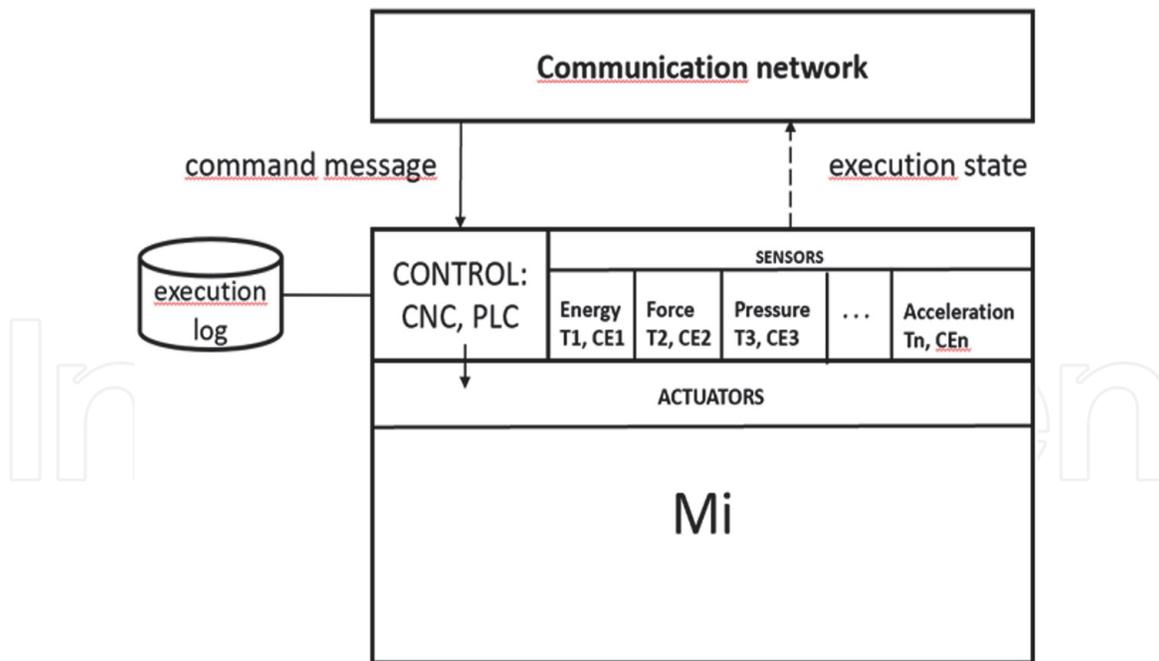
Physically, we assume that each source message emitted by a machine tool is in a semi-structured text format (a sort of simplified JSON): that is, it is a succession of text fields, divided by a field terminator. This provides simplicity and flexibility with the greatest possible space savings. It is possible to store anything within the main dataset, as long as each data has the same information placed in the same order, with the same data type format; otherwise it will be necessary to carry out a pre-processing of the source messages before storing them, downloading the data that do not conform to the expected format in a separate archive.

### 3. System description

#### 3.1 The model for machine tool

Manufacturing systems organize machine tools, material handling equipment, inspection equipment, and other manufacturing assets in a variety of layouts. With the advent of Industry 4.0 technologies, these manufacturing resources can be networked, and cyber-physical manufacturing systems can be implemented that integrate hardware and software with mechanical and electrical systems designed for a specific purpose. The model of **Figure 2** shows a representation of the main components of a generic industrial machine from the point of view of data collection and is sufficiently general to be applied to many industrial machines that can be part of process, cellular and line layouts.

The generic machine  $M_i$  receives from the network a command message known as part program. It is a set of detailed step-by-step instructions executed by the CONTROL system (CNC, PLC) that direct the actions implemented by the actuators. The actuators act as transducers changing a physical quantity, typically electric current, into another type of physical quantity such as rotational speed of an electric motor. During the execution of a part program, the machine tool operates and the CONTROL system produces a log containing data about the executed instructions as well as control messages that indicate some particular machine state. In the meantime, execution data captured by the sensors are sent via the communication



**Figure 2.**  
A simple model for machine tool data collection.

network to the factory control system. Here, the feedback loop is closed, and feedback actions can eventually be taken.

The generic sensor is represented by its two constituent parts: the transducer ( $T_i$ ) and the control electronics ( $CE_i$ ); this distinction is useful because even if the sensor is a unified whole, the transducer and the control electronics can be placed into different area of the machine tool. For example, the accelerometer transducer can be placed on the spindle while its control electronics is usually placed within the cabinet together with other control subsystems.

### 3.2 System functionality

The event-based dynamic cyber-physical system proposed here integrates physical devices with advanced Text Mining analytical technologies and is very general, therefore adaptable to any production domain. It needs the ontology of the messages emitted by the data sources of a machine tool during its normal operation and is able to intercept its (even slightly) anomalous behaviors, which allow to evaluate whether it is moving towards a failure state such as to require the activation of safeguard procedures.

The system consists of a design pre-processing phase to create the main message ontology and which is performed only once for each data source (for example a sensor), and an algorithmic runtime phase. Each message correspond to an event of the data source, has the form of a text messages and has associated an adequate alert level.

### 3.3 Pre-processing design phase

The pre-processing phase is performed only once for each data source and allows you to create the initial ontology of the messages. It consists of four main steps.

In the first step of the pre-processing phase, for each data source the set of all messages that it can emits is identified and normalized. In particular, for each industrial machine and for each data source associated with that machine, the possible types of messages that can be issued must be identified; then, these data

must be interpreted, choosing only the pertinent ones; finally, the data must be semantically normalized, to adapt it to a common data semantics.

In the next step of the pre-processing phase, the system associates an alert level, based on a chromatic scale, to each of the messages coming from the previous step. The levels are as follows:

**White** Level (no problem).

**Yellow** Level (warning): There have been sporadic, but not serious, anomalies and the system is able to continue without problems.

**Orange** Level (serious): some serious anomalies, or some cluster of anomalies, have been found in the system, which could affect the future of work.

**Red** Level (very serious): Very serious anomalies were found in the system, capable of affecting, in the very near future, the production activity.

**Black** Level (immediate stop): the system runs the immediate risk of irreparable equipment or product failures.

In the third step of the pre-processing phase, to be carried out only if there are several data sources associated with the machine tool, composite messages are created, obtained by composing the messages with at least an yellow alert level, a composite message. The order in which messages are juxtaposed is predefined and must be the same as it will appear at runtime. At this point, a general alert level is associated with each composite message.

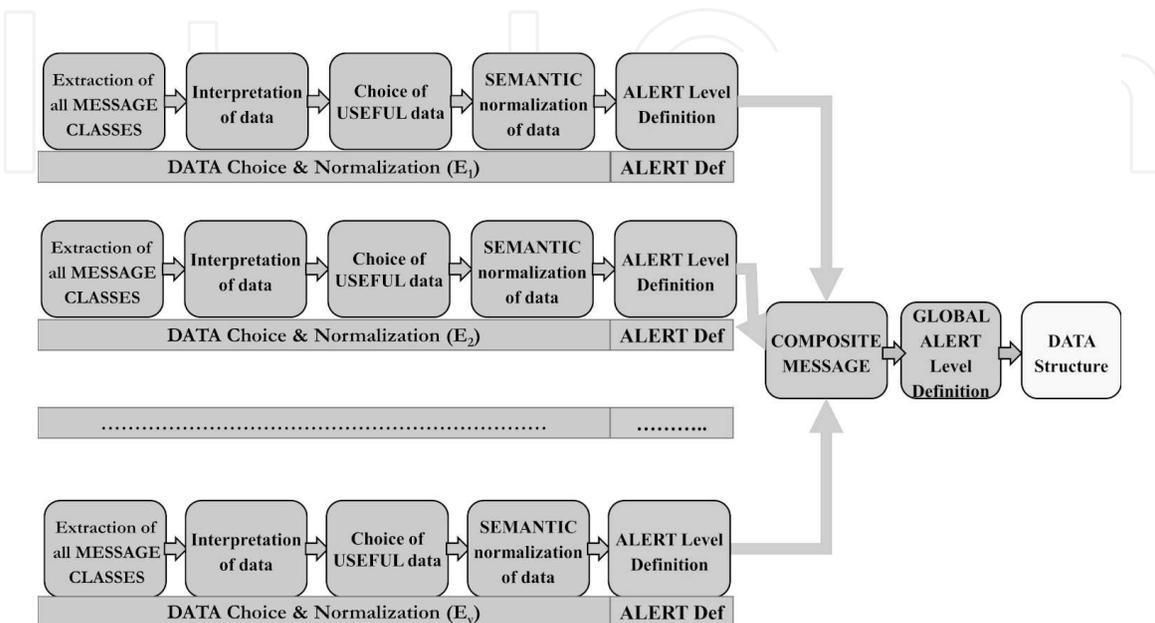
Finally, a digital data structure is built containing all composite messages: a text dictionary, in which each line contains a composite message and the relative general alert level.

Steps 1 and 2 must be performed for each of the data source entities associated with a specific machine tool. Identifying alert levels requires the help of the machine tool expert, thus steps 2 and 3 need the help of a machine tool expert.

The whole data design process is reported in **Figure 3**.

### 3.4 Runtime phase

In the runtime phase, the real data coming from a machine tool are collected and normalized according to the specifications identified in the pre-processing phase. Then, all messages with a compatible timestamp are aggregated into a line of text to



**Figure 3.**  
 Data design process.

form a single composite message that has the same layout as the composite messages in the dictionary. The layout of a composite message from a machine tool and  $n$  Data Sources, is as follows:

Machine ID + Timestamp.  
+ DS<sub>1</sub> ID + DS<sub>1</sub> Message.  
+ DS<sub>2</sub> ID + DS<sub>2</sub> Message.  
+ ...  
+ DS <sub>$n$</sub>  ID + DS <sub>$n$</sub>  Message.

The algorithms look up this compound message in the dictionary of all compound messages and extract its global alert level. They identify all occurrences of all messages in the text, even if they are overlapped (partially or completely).

At this point the system analyzes the alert levels found, their frequency and, possibly, their relative positioning both in time and in space, and predicts possible future malfunctions.

The system implements different types of analysis algorithms. The simplest algorithm catalogs the entire cluster of messages and identifies the warning levels that appear most frequently. A more sophisticated algorithm analyzes smaller homogeneous clusters of these composite messages. Another algorithm identifies the presence of sub-clusters, even small ones, which indicate a possible future malfunction of the machine.

The software is written in DELPHI (Visual Development Environment), is highly parameterized (to allow quick modification) and is composed of about 2,000 lines of code structured in 30 operating modules, different methods for managing the interface and uses a large number of predefined libraries.

### **3.5 Detailed algorithms description**

The system at runtime is fast and accurate in identifying possible anomalous situations of a machine tool. It consists of a pre-processing step (to be performed only once, when the program starts) which has the purpose of building the entire data structure (the dictionary of all messages with their relative alert level) and an actual processing step.

The system is independent on the dictionary size. It reads the input text, consisting of one or more messages, one character at a time and scrolls through a finite automaton; when it encounters a final state (corresponding to the end of a message), it emits the alert level associated with that message. The number of state transitions is proportional to the number of characters read in input.

The steps are:

**Pre-processing.** The system builds the data structure that the algorithms will use when running. Pre-processing only needs to be done once, at startup. In particular, the system constructs, in the central memory, a finite state automaton from all the elements of the dictionary, and its behavior is completely determined by a (small) set of states and by some simple functions.

**Matching.** The finite state automaton continuously reads the messages from the input log file, character by character. When it reaches a final state, the automaton shows the alert level of the message corresponding to this state. The algorithm is able to identify all occurrences of all messages, even if they were partially or totally overlapping. Lines with no messages (i.e. with “blank” content) are ignored.

**Analysis.** In this step, the system analyzes a set of alert levels identified in the previous phase. Normally, it analyzes a group of alarms, usually those issued in a period; then, based on the frequency and “severity” of the alerts extracted, it makes

a classification of the health of the machine. The system warns if the analysis of the message cluster indicates a possible future malfunction of the machine tool. The system can also provide an immediate response if a single alert is very critical.

### 3.6 Performance analysis of algorithms

The pre-processing step (i.e. the construction of the FA) requires a time linearly proportional to the sum of the lengths of all the messages present in the dictionary, i.e. the total number of characters in the dictionary.

The matching algorithm for a set of textual log messages with a total length of  $k$  characters requires  $n$  state transitions ( $n \leq 2k$ ). Therefore, the analysis of a message takes a linear time with respect to the number of characters, and this is a lower limit, whatever the algorithm used among those who read the message character by character, since all characters of the message must be read.

Algorithms with an approach different from the one proposed here, should refer to the method that considers a whole input message as a single entity. These methods look for a message in the set of all possible messages (the dictionary). From the literature [15], we know that no algorithm in that class can use less than  $O(\log n)$  steps to look up a single message in a dictionary of  $n$  messages.

Therefore, the algorithms used here are independent of the size of the dictionary, which can be as large as we want without worsening the search time, while the classical algorithms are dependent on the size of the dictionary: the larger it is, the more time it takes to search for a message within it.

The differences are even greater by admitting the possibility of editing the dictionary. With our approach, the dictionary does not need to be sorted, so adding, changing, or deleting one or more dictionary entries is done at virtually no computation cost. On the other hand, with the classical approach, the dictionary must be kept orderly. Therefore, in case of cancelation, insertion or modification even of a single entry, the dictionary must be reordered, and this costs at least  $O(\log n)$  operations, possibly with physical movement of the entries from one memory area to another.

Another advantage of our approach is of a technical nature: the algorithms are all executed in central memory, while a classical method largely uses secondary memories (which are much slower by several orders of magnitude).

In addition, each log message (dictionary entry) is made up of (many) words and other non-alphabetic symbols and this means that the dictionary size can be very large and, furthermore, the search algorithms need the messages to be well delimited and not superimposed, while our approach is able to identify even totally overlapping or non-delimited messages and present within plain text sentences.

## 4. A simple case study

We have also implemented a prototype of the cyber-physical system presented in this paper. In this section we report the results of some preliminary tests we conducted on some data coming from a machine tool currently operating in an important company, operating in Southern Italy, which produces metal molds for other national and international companies.

In particular, we analyzed a log file relating to a period of 194 hours of continuous work of a machine tool, from 4:25 on 13 / Feb / 2019 to 21:34 on 20 / Feb / 2019, in which it issued approximately 300,000 log messages.

In the preliminary testing phase, we used data from a single data source from a single machine tool, then we considered simple, not composite messages, because our initial interest was to verify the feasibility of algorithmic message search in ontology.

## 4.1 Case study data description

A log is the sequential and chronological recording of the operations and events coming from a specific industrial machine tool. Log messages are stored in text format in one or more files, one record per line with each line containing only one message. Generally, these registrations are done in an automated way. Each record stores everything that happens on the machine, so a log file holds both information about normal machine operation and about errors and problems or even slight deviations from the norm.

This section shows the actual data relating to the case study examined, but the information contained in a log file of a generic machine tool is approximately the same, regardless of the machine; at most it could slightly change the data format in each individual field or their mutual position in the log file.

Therefore, the case study presented here is representative of many industrial machines.

Here, there is an example of the industrial machine log file from the case study.

```
13/02/2019 04:25:24;MSG_SYS;Scrive;Fine corsa asse., Y+;
13/02/2019 04:25:26;MSG_SYS;Scrive;E 2034:indice variabile errato in riga PLC..
4239;
13/02/2019 04:25:28;MSG_SYS;Scrive;Fine corsa asse., Y+;
13/02/2019 04:25:30;MSG_SYS;Scrive;E 2034:indice variabile errato in riga
PLC.. 4239;
```

The log file has four blocks of useful data and a fifth block that is not useful for analysis. Each field is separated from the next by the semicolon symbol (;). The essential information contained in the first four blocks of the first row (the one in bold) of the example above are:

13/02/2019 04:25:24;	<b>Date/time of recording of the event</b>
<b>MSG_SYS;</b>	Process PID, i.e. the identification of the running process,
<b>Scrive;</b>	The operation done
<b>Fine corsa asse., Y+;</b>	The status or result of the execution of the event

As described in the following section, we have extracted all the different messages from the 300,000 input lines; therefore we have assigned, with the assistance of the machine tool technician, an alert level to each single different message.

A single log record stored in the dictionary is a line containing the following information, separated by an hashtag:

**“Machine ID” + Date/Time # Process PID # Oper. Done # Result # “Warning Level” (1).**

Machine ID is a code that allows to identify the single machine. Since the same message can be emitted several times on the same machine, but never at the same instant, each message is identified by ID + Date/time. Each message corresponds to a single fact and, associated with each message, there is the level of warning issued. By adding the ID of the data source to the message, it is possible to integrate messages from different data sources that insist on the same machine.

## 4.2 System prototype run and results

In the preliminary phase, we had to identify the different types of messages emitted by the machine for which we had the log files. We analyzed 300 log files, each containing 1000 messages, from 4:25 on feb/13/2019 to 21:34 on feb/20/2019.

```

    C:\form1
    START SOURCE ONLY WORDS COMPLETE Stat 16 sep 2020 - UNISA DISA-MIS PRINT EXIT
    MINING - Rel. 2.0 (16 sep 2020) - Test performed on: 27/09/2020 18:41:53
    ***** GLOBAL ANALYSIS *****
    Total Messages found: 299998

        WHITE 299039 99.68033%
        YELLOW 561 0.18700%
        ORANGE 396 0.13200%
        RED 2 0.00067%

    Alerts not white: 959
    Alerts not white AND not yellow: 398
    Alerts RED and BLACK: 2
    ***** MESSAGES *****
    Number of different messages: 1499
    Occurr. ALERT MESSAGE
    -----
    6 *** ORANGE *** analisi_blocco;azzera;blk src: c2x0y0 --- blk dest: passato a mc;
    2 *** YELLOW *** analisi_blocco;blocco_singolo;blk src: m06t5 --- blk dest: passato a mc;
    1 *** YELLOW *** analisi_blocco;blocco_singolo;blk src: m06t6 --- blk dest: passato a mc;
    3 *** ORANGE *** analisi_blocco;blocco_singolo;blk src: m9063 --- blk dest: passato a mc;
    1 *** ORANGE *** analisi_blocco;blocco_singolo;blk src: m963 --- blk dest: passato a mc;
    1 *** YELLOW *** analisi_blocco;blocco_singolo;blk src: x-535.505y-187. 132r --- blk dest: passato a mc;
    1 *** YELLOW *** analisi_blocco;blocco_singolo;blk src: y160r --- blk dest: passato a mc;
    2 *** YELLOW *** analisi_blocco;blocco_singolo;blk src: z48r --- blk dest: passato a mc;
    2 *** ORANGE *** gestione_file;apre file in edit;d:\s4000\partprog\19001-10\fori08;
    19 *** YELLOW *** gestione_file;apre file in edit;d:\s4000\partprog\c-18507-50\c-18507-50-matrice_p2_2_1-cd7
    4 *** YELLOW *** gestione_file;apre file in edit;d:\s4000\partprog\c-19001-10\lato b\p35c2c161b.sel;
    3 *** YELLOW *** gestione_file;apre file in esecuzione grafica;d:\s4000\partprog\19001-10\c-19001-10-p17-f4
    3 *** YELLOW *** gestione_file;apre file in esecuzione grafica;d:\s4000\partprog\19001-10\dany;
    3 *** YELLOW *** gestione_file;apre file in esecuzione;d:\s4000\partprog\c-18507-50\c-18507-50-matrice p2 2
    
```

**Figure 4.**  
 One week summary of log message analysis.

Of the 300,000 messages (the non-empty ones are actually 299,998), which we have inserted in a single file, those with different “semantic” content are 1,499. To achieve this we have eliminated every message that does not have useful information and the first 20 characters (date and time) of each remaining line and then we have identified all the different messages. At the end of this first pre-processing phase, we have a file that contains all and only 1,499 different messages.

In the next step, with the assistance of the machine tool technician, we assigned an ALERT level to each of the 1,499 different messages. The alert levels are ordered according to the increasing level of severity: white, yellow, orange, red and black.

The final document contains the associations between the 1,499 messages and the related warning level. We report here the screen of the test sent by us on the log file and the first part of the list of messages encountered with the relative multiplicity and alerts: In **Figure 4** there is the final report of a week’s analysis of log messages from a real industrial machine.

During this week there were three times when groups of messages occurred that required the supervision of experienced personnel, but none of these alarms turned into a request to stop the machine. Two interventions were due to a red alert and another due to an anomalous aggregation of orange and yellow alerts in a short period. At the end of the week, during a planned machine downtime, some adjustments were made, suggested by the presence of some clusters of messages relating to a slight deviation of the machine performance, compared to the estimated one.

Since the messages can be analyzed in real time, if clusters of “dangerous” messages occur during the operation of the machine, the machine experts would be able to intervene in time to prevent irreparable damage. Note that the message cluster can also be formed by several messages of mild severity issued, however, in an anomalous configuration or within very short times.

## 5. Conclusions

We have presented here an innovative methodology and an associated fast and efficient software system prototype, for the algorithmic prediction of industrial machine tools malfunctions, adaptable to any type of company. It integrates

machinery and physical devices with the analytical technologies of Text Mining and allows the identification of anomalous behavior of a machine tool, even of minimal entity, rarely perceived by other strategies.

The system performs its analysis without waiting for the end of the shift or a machine stop. After recognition, it can initiate automatic safeguard procedures, call a human expert, or schedule some minor tuning operations. The system works without waiting for the shift to end or the machine to stop.

The algorithms require linear execution time on the number of input characters, run on a data structure completely on RAM and are independent on the data structure size, which can be modified without actual computational costs, as it is not sorted. A classic approach, on the other hand, requires searching for a message in a set of possible messages using efficient algorithms, which work largely on secondary memories and depend on the size of the data structure that need to be, necessary, sorted, and therefore it takes time, not irrelevant, to add, modify or delete an entry in it, possibly by physically moving items from one memory area to another.

Last, but not least, is the fact that a classic approach is inadequate because a log message is made up of many words and others non-alphabetic symbols and the data structure size could be very large.

We believe that this approach can bring significant competitive advantages to a company in which the effective and precise predictive analysis of machine tools is a necessity to be pursued by spending as little time as possible, obtaining as precise a result as possible, limiting false recognition errors, as much as possible.

## Author details

Giancarlo Nota<sup>†</sup> and Alberto Postiglione<sup>\*†</sup>

Department of Business Sciences - Management and Innovation Systems  
(DISA-MIS), University of Salerno, Salerno, Italy

\*Address all correspondence to: [apostiglione@unisa.it](mailto:apostiglione@unisa.it)

<sup>†</sup> These authors contributed equally.

## IntechOpen

© 2021 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

## References

- [1] Löfsten H. Management of industrial maintenance – economic evaluation of maintenance policies. *International Journal of Operations & Production Management*. 1999 Jul;19(7):716–737. Available from: <https://doi.org/10.1108%2F01443579910271683>.
- [2] Ahmad R, Kamaruddin S. An overview of time-based and condition-based maintenance in industrial application. *Computers & Industrial Engineering*. 2012 Aug;63(1):135–149. Available from: <https://doi.org/10.1016%2Fj.cie.2012.02.002>.
- [3] Carneiro D, Nunes D, Sousa C. A Decision-Support System for Preventive Maintenance in Street Lighting Networks. In: 18th International Conference on Hybrid Intelligent Systems (HIS 2018), Porto, Portugal, December 13–15. vol. 923. Springer International Publishing, AISC series; 2019. p. 272–281.
- [4] Vilarinho S, Lopes I, Oliveira JA. Preventive Maintenance Decisions through Maintenance Optimization Models: A Case Study. *Procedia Manufacturing*. 2017 Jun;11:1170–1177.
- [5] Szczepaniak M, Trojanowska J. Preventive Maintenance System in a Company from the Printing Industry. In: et al IV, editor. *Advances in Design, Simulation and Manufacturing II*. DSMIE 2019, Lecture Notes in Mechanical Engineering. Springer International Publishing; 2019. p. 351–358.
- [6] Usher JS, Kamal AH, Syed WH. Cost optimal preventive maintenance and replacement scheduling. *IIE Transactions*. 1998 Dec;30(12):1121–1128.
- [7] Li J, Tao F, Cheng Y, Zhao L. Big data in product lifecycle management. *The International Journal of Advanced Manufacturing Technology*. 2015;81: 667–684.
- [8] DeFelice F, Petrillo A, Monfreda S. Improving Operations Performance with World Class Manufacturing Technique: A Case in Automotive Industry. In: *Operations Management*. InTech; 2013. p. 1.
- [9] Postiglione A, Monteleone M. Towards automatic filing of corpora. In: *Proceedings of 18'eme COLLOQUE INTERNATIONAL "Lexique et Grammaires Comparçs"*, Parco Scientifico e Tecnologico di Salerno e delle aree interne della Campania, Salerno, Oct 6–9; 1999. p. 1–9.
- [10] Elia A, Vietri S, Postiglione A, Monteleone M, Marano F. Data mining modular software system. In: Arabnia HR, Marsh A, Solo AMG, editors. *SWWS2010 - Proceedings of the 2010 International Conference on Semantic Web & Web Services*. Las Vegas, Nevada, USA, Jul. 12–15. CSREA Press; 2010. p. 127–133.
- [11] Elia A, Postiglione A, Monteleone M, Monti J, Guglielmo D. CATALOGA: a software for semantic and terminological information retrieval. In: Akerkar R, editor. *Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS 2011*, Sogndal, Norway, May 25–27, 2011. ACM Press; 2011. p. 11.
- [12] Elia A, Postiglione A, Monteleone M. CATALOGA: a software for semantic-based terminological data mining. In: *1st International Conference on Data Compression, Communication and Processing*, IEEE, Palinuro (SA), June 21–24. IEEE Computer Society; 2011. p. 153–156. Available from: <http://www.computer.org/csdl/proceedings/ccp/2011/4528/00/index.html>.
- [13] Postiglione A, Monteleone M. Semantic-based bilingual text-mining. In: *Second International Conference on*

Data Compression, Communication, Processing and Security (CCPS 2016), September 22–23, Cetara (SA); 2016. p. 1–4.

[14] Postiglione A, Monteleone M. A Linguistic Semantic Text-Mining for Multiword Units. In: Roni R, editor. MANTUA HUMANISTIC STUDIES. vol. XII. Mantua Humanistic Studies, Mantova:Mantova: Universitas Studiorum; 2020. p. 445–459.

[15] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms, Third Edition. 3rd ed. The MIT Press; 2009.

IntechOpen