Finding meaningful representations of SCADA-log information for data-driven condition monitoring applications

Conference Paper · December 2020

DOI: 10.5281/zenodo.5467486

1 author:

Simon Letzgus
Technische Universität Berlin
20 PUBLICATIONS **116** CITATIONS

SEE PROFILE

16th EAWE PhD Seminar on Wind Energy
14-16 December 2020
Online Event

Finding meaningful representations of SCADA-log information for data-driven condition monitoring applications

Simon Letzgus[a,b]

[a]Technische Universität Berlin, Machine Learning Group

E-mail: simon.letzgus@tu-berlin.de

Keywords: wind energy, SCADA log data, condition monitoring, NLP, machine learning

1 Introduction

Analysis of data from supervisory control and data acquisition (SCADA) systems to monitor wind turbine condition has attracted considerable research interest in recent years. Most approaches utilize time series data from sensors placed all over the turbine and apply methods from the machine learning (ML) domain for early failure detection (compare [1]). Besides, SCADA systems usually produce log-files which contain information about operation conditions or control events but also warning and alarm messages in case sensors measurements come close to or exceed pre-defined limits. In the condition monitoring context, this data is also of high value and has been used to filter and annotate sensor time series [2], identify message-patterns related to failures ([3], [4], [5]) and predict unplanned stoppages directly from the log-messages ([3]). However, most of these approaches use time-sequence or probability-based analysis rather than ML methods. One reason is that a ML model usually requires a numeric vector representation of fixed size as input. The challenge of finding such representations for symbol sequences of variable length is well known in the ML domain, especially in natural language processing (NLP). Within this study, we extend the Correlated Occurrence Analogue to Lexical Semantic (COALS) algorithm by incorporating temporal information (COALS-t) and apply it to SCADA log-data. We demonstrate that the method can find meaningful representations of SCADA log messages and message sequences. This enables the application of off-the-shelve ML methods. The approach will be illustrated using multiple years of operational SCADA data from several turbines. The remainder of the paper introduces the SCADA data set (section 2) and the methods being applied (section 3) before presenting and discussing the results (sections 4 and 5).

2 SCADA log data

Format and content of SCADA log files vary for different manufacturers and systems. What they have in common, however, is a tabular structure in which operational events are logged in temporal order. Each event is assigned a starting time which often is accompanied by additional temporal information, such as an event end time or the time of an event-related turbine restart. Apart from the mandatory temporal information, events are usually characterized by an event identifier, mostly a system-specific multi-digit event code, and a short event description which gives high-level information about the nature of the event and therefore helps operators, maintainers and analysts to partially decode the cryptic event-ids. Other columns of the log file might contain a classification of the event into different severity categories (information, warning, alarm, fault), assign the event to a certain turbine sub-assembly or provide information on the current operational status of the turbine. Due to the wide range of potential log configurations and formats this study will focus on representations derived from basic information which can be found across almost all SCADA log files, namely start-timestamp, message-ids and message description.

For the analysis, we use data SCADA data from a wind farm consisting of 5 geared 2 MW turbines with asynchronous generators. Over 2 years 313,590 log messages were generated by the wind farm's

Turbine	A	B	C	D	E	Farm
Total messages	76,485	54,068	61,014	60,983	61,040	313,590
10min avg	0.73	0.51	0.58	0.58	0.58	2.98
10min max	179	94	223	601	66	610
Unique messages	165	184	154	226	150	281

Table 1: Summary statistics of wind farm log count

Timestamp	ID	Description
2020-12-11 10:23:45	m_005	high temperature tX: X°c
2020-12-11 11:23:50	m_097	pause Xkw Xrpm
2020-12-11 11:23:50	m_157	extra info. err:309
2020-12-11 11:28:50	m_055	restart

Table 2: Exemplary excerpt from log files.

SCADA system which corresponds to an average of around 3 messages per 10min interval (compare Table 1). However, in some 10min intervals, up to 610 messages were triggered. These numbers are in line with earlier count-based analysis of WT SCADA logs (compare [4]). The log data format contains a single (start) timestamp and a description of each message from which a unique message-id can be derived (compare Table 2).

3 Methods

We first introduce the NLP tools used in this study, then discuss their application to the given data format. Therefore, we will generally talk about the representation of log-instances and afterwards emphasize the different implications for event ids or event descriptions. Finally, we introduce some metrics to evaluate the representations.

3.1 Methods to find log-message vector representations

A rather straight forward representation of a non-numeric data instance is a so-called one-hot-encoding. For a fixed set of unique instances $M = (m_1, m_2, ..., m_N)$ each instance is represented by a binary N-dimensional vector $R_{\text{OH-m}}$ with all but one entries being set to zero. The non-zero entry at position i uniquely identifies the instance m (compare (1)). This generally enables the application of ML-algorithms and it has been applied in the context of SCADA log processing (e.g. in [3] to detect pitch faults based on SCADA-logs and in [5] for alarm clustering). However, the method creates overly sparse high-dimensional representations which at the same time do not contain any information about relations between instances. This points towards a powerful concept in NLP which is to describe an instance with the help of its surrounding.

$$R_{OH-m} = [r_1, r_2, ...r_N] \quad \text{with} \quad r_i = \begin{cases} 1 & \text{if } m = m_i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Within this study, we adjust and apply the Correlated Occurrence Analogue to Lexical Semantic (COALS) algorithm [6], a method from the Latent Semantics Analysis (LSA) family. It extracts vector representations by singular value decomposition (SVD) of a normalized instance co-occurrence matrix (compare 1). We propose to incorporate the temporal information contained by each log entry's timestamp into the co-occurrence matrix by adding up processed timestamp differences between neighbouring log messages instead of the commonly used counts (compare Figure 1). The inversion of the time-difference keeps the co-occurrence matrix intuitive with large entries representing high similarity (compare eq. (2)). Additionally, the parameter ϵ prevents numeric instability and acts as a bandwidth parameter which controls for how aggressive time differences are taken into account. Also, this allows for the window size to be selected rather large because the time-differences serve as a natural weighting factor (in NLP the window size is often a one-digit number since sentences usually don't stretch across thousands of words). The subsequent normalization and decomposition procedure remains as described in [6]. For the remainder of the paper, we refer to this procedure as COALS-t.

$$\Delta count_t = \frac{1}{\Delta t_{sec} + \epsilon} \tag{2}$$

Timestamp	Message
2020-12-14 10:07:03	m_1
2020-12-14 11:01:15	m_2
2020-12-14 11:23:50	m_3
2020-12-14 11:23:55	m_4
2020-12-14 20:31:17	m_3
2020-12-14 20:31:22	m_4

Window count

Center	m_1	m_2	m_3	m_4
m_1	0	1	0	1
m_2	1	0	1	0
m_3	0	1	0	3
m_4	1	0	3	0

Count-based co-occurance

Window times *

*10^4	m_1	m_2	m_3	m_4
m_1	0	2	0	0
m_2	2	0	4	0
m_3	0	4	0	20
m_4	0	0	20	0

Time-based co-occurance

* compare $\Delta count_t$ in equation (2) with ε = 1000

Figure 1: Exemplary sequence of log messages (left) with construction of traditional count-based co-occurance matrix (center) and the proposed time-based co-occurance matrix (right).

3.2 Methods to find log-message sequence representations

In NLP series of words add up to sentences and then to documents which represent natural sequence structures to be analysed. In the log-analysis context, the question of how to properly define the length of a log-sequence is less trivial. When having in mind potential fusion of SCADA sensor and log data 10min intervals seem to be a reasonable choice allowing simple temporal synchronization between the two data formats. For other applications, event-based sequence selection might be more suitable (compare e.g. [5]). While the optimal choice of sequences will always be problem-specific, the methods to transform them into vector-representations are mostly independent from the actual sequence length.

A relatively simple but often sufficient representation of instance sequences is the so-called bag-of-words (BOW) or term-frequency approach. The sequence representation R_{BOW} is constructed by simply summing up all vector representations R_k of a sequence $S = (R_1, R_2, ...R_K)$ (compare (3 - left)). Despite ignoring the order of instances the method was often reported to be surprisingly effective, especially in text classification [7]. For our application both, the one-hot as well as COALS-t representations can be processed this way. An extension to the idea of a simple BOW sequence is to weight the individual instance representations R_i with their inverse sum of overall occurrences N_{Rk} (compare 3 - right). This incorporates the idea from information theory that rare instances contain more information and are therefore weighted higher. Such term-frequency inverse-document-frequency (TF-IDF) representations have been successfully used in the context of information retrieval and will be applied for log-sequences within this study.

$$R_{BOW} = \sum_{k=1}^{K} R_k \qquad \text{and} \qquad R_{TFIDF} = \sum_{k=1}^{K} \frac{R_k}{N_{Rk}} \tag{3}$$

3.3 Evaluation of log-representations

Before applying the individual NLP tools used in this study, lets quickly emphasize the goals we have in mind when applying them. The aim is to construct a dictionary which maps each log-instance or log-sequence to an M-dimensional real-valued vector. Additionally, we aim for the vector to implicitly contain information about the nature of the respective instance which facilitates solving a downstream task. This means one way to evaluate each set of representations is to evaluate their corresponding performance on the specific task. However, in this study, we discuss representations generically which makes such an extrinsic evaluation infeasible. Therefore, we focus on so-called intrinsic evaluation which tests the representations for coherence with our intuition. In our example, this would mean that syntactically or semantically similar events are grouped closer together in vector space than their unrelated counterparts. Put concretely, we evaluate log-message-representations for the intuitive principles shown in Table 3 and compare distances between those grouped examples against distances to other message representations. For the evaluation of the log-sequence representation we rely on the same intuition: we assume that message sequences that proceed the same error message look more similar than the sequences proceeding other error messages. We therefore evaluate and compare the similarity of sequences consisting of the messages triggered in a 10-minute window before the respective error message.

Similarity in vector space is measured utilizing different distance metrics which can be applied to both, log-embedding and log-sequence-embedding evaluation. For binary comparisons of representations,

Table 3: Examples for different categories of SCADA log-message similarity

	Syntactic twins	Syntactic / semantic sibilings	Semantic cousins
Description	messages almost identical	message body with major similarity	syntactically different but shared root
Example	igbt pump off igbt pump on	feedback = 0, hydraulicmotor feedback = 0, gearoil cool feedback = 0, nacelle fan	event service state 0 keq switch remote control pause pressed on keyboard

we apply the popular cosine similarity (compare 4). Additionally, we project the high dimensional vector representations into a two dimensional plain which allows for visualization and interpretation of the respective distances between individual instances. It must be noted that during the dimensionality reduction some information will be lost but the most prominent characteristics are usually preserved. In this study we apply t-SNE, a non-linear dimensionality reduction technique designed to preserve similarity information from the high dimensional space [8].

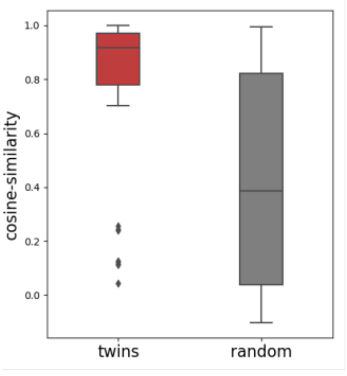
$$similarity(R_a, R_b) = \frac{R_a * R_b}{||R_a|| * ||R_b||} = \frac{\sum_{i=1}^{n} R_{ai} * R_{bi}}{\sqrt{\sum_{i=1}^{n} {R_{ai}}^2} * \sqrt{\sum_{i=1}^{n} {R_{bi}}^2}} \tag{4}$$

4 Results

4.1 SCADA-log messages representations

In this section, we focus on the analysis and comparison of the log-message representations. Since one-hot representations do not capture any inter-message relations we focus on the representations yielded by the COALS-t method. We chose a 10-dimensional vector representation obtained with a window size of 100 to either side of the central log-message and a temporal bandwidth ϵ of 1000.

Firstly, we will look at the representations of messages from the syntactic twins category. It must be noted that the method was not given any syntactical information about the messages itself. Similar vector representations of syntactically close messages emerge from the fact that they are likely to appear in a similar context. We automatically group all messages which syntactically differ by not more than 2 characters. Then we calculate the binary cosine-similarities within each of the 26 sub-groups, which contain up to four highly similar messages (quadruplets, if you will). As a result, we obtain 48 intra-group distance values. For their better interpretation, we compare them to an equal amount of binary cosine-distances between random messages. The results are shown on the left side of Figure 1. Messages of the twin category are indeed grouped much closer to each other in vector space than their randomly selected counterparts.



Message	pitch a ref: X°, act.:X°	cos-sim		restart	cos-sim
Closest neighbours	pitch dev. min: X° max: X°	0.994		stop Xkw Xrpm	0.940
	pitch b ref: X°, act.:X°	0.992		too many auto-restarts X	0.911
	ovphwerr udc Xv state X	0.990		ch active d load cX X%	0.909
Error	error_144			error_725	
Closest neighbours	high windspeed: X m/s	1.0		no rt, high rotor cur Xa	1.0
	yaw position is changed: X°	0.987		no rt, rotor hw.error Xa	0.998
	yaw prot. Xm/sX wdir X°	0.979		no rt ex.hig.rot.cur Xa	0.997

Figure 2: Left: comparison of similarity between twin and random messages. Right: examples of most similar messages to selected messages (top) and errors (bottom).

Secondly, we compare messages from the syntactic/semantic siblings and semantic cousins category by visualizing their two-dimensional t-SNE representations relative to all other messages (compare Figure 3). The top row presents three syntactic/semantic sibling examples where all message representations that contain the respective title words are highlighted. The bottom row presents semantic message clusters, such as the *Thermal* category which groups message names containing *high* and *temperature* or *ventilation* (or their respective abbreviations). The *Service* category contains messages connected to physical on-site visits (examples can be found in Table 3). The plots show that coherent message clusters connected to a particular frequent event (cable twist example), prevailing condition (thermal example) or general status (service example) are formed. Moreover, their relative position is partially intuitive as well. The cable-twist related messages for example can be found close to the clusters where most of the yaw-related messages are located indicating their causal relation. Finally, the *Error* example visualizes the relative location of error messages. Many can be interpreted by associating them with a nearby error message or the theme of their cluster (also compare Figure 2 -right/bottom). Note, that in the main region of service messages there is no related error messages indicating that service activities are not directly related to a specific error.

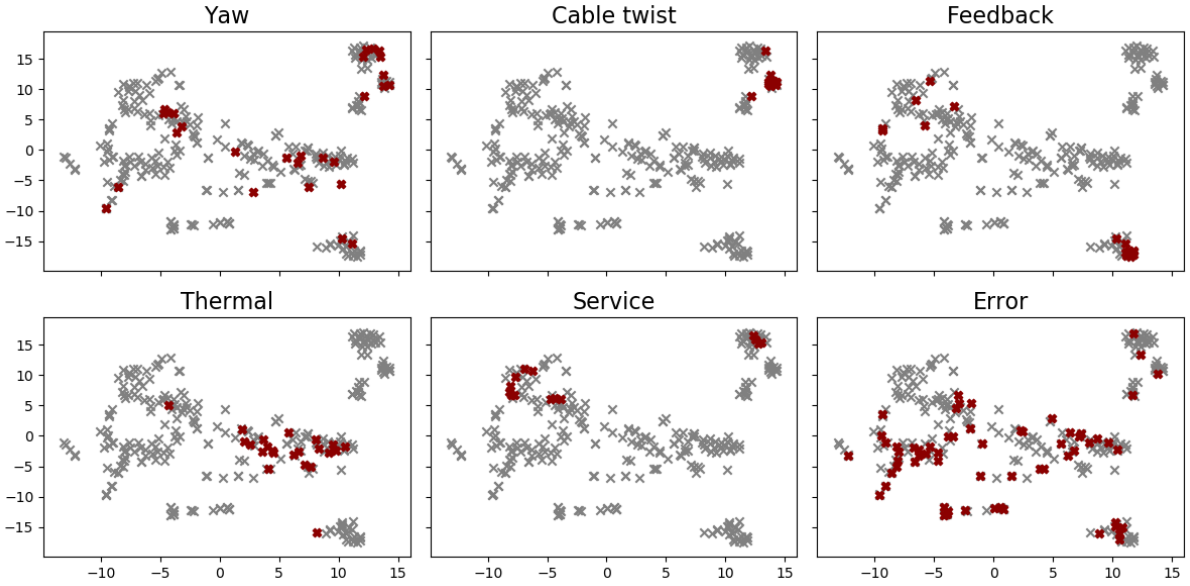


Figure 3: Visualization of t-SNE projections of message representations grouped by intuitive themes (red) in relation to all other messages(grey).

4.2 SCADA-log sequences representations

To construct the SCADA-log sequence representations we combine both presented methods of log-message encoding (one-hot and COALS-t) with either the BOW or the TF-IDF approach. This results in 4 different representations for each of 1341 sequences proceeding one of the 62 unique error messages. Since we want to run binary similarity comparisons we first had to exclude a few sequences connected to errors that occur only once during the recorded time. Then we compute the binary cosine-similarity within each error category. This means that if an error occurs five times in the data set we will compute 10 binary distances. This results in approximately 65,000 binary similarity measures for the same error-type analysis. To facilitate interpretation we calculate the same amount of distances in between sequences proceeded by different error types. The results in Figure 4 show that all configurations can group sequences proceeding the same error message closer together than their random counterparts. The one-hot configurations are characterized by a much stronger dissimilarity between randomly chosen sequences. COALS-t embeddings can minimize the distance between similar sequences, but the difference to randomly chosen sequences is not as pronounced. This might originate from the fact that the error messages themselves form clusters

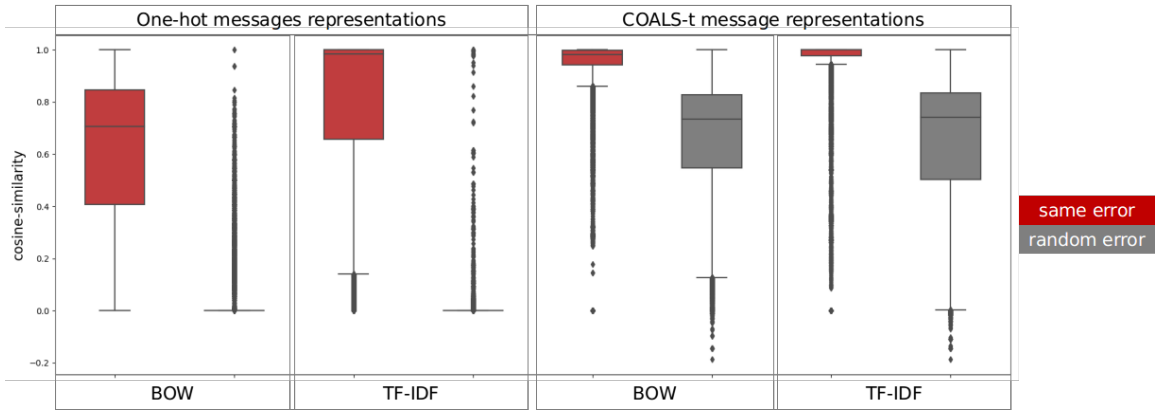


Figure 4: Cosine distances for sequences proceeding the same error messages (red) and random error messages (grey) for different combinations of log-message- and sequence-representations.

and a random choice does not ensure a large distance between them (compare Figure 3). When comparing the sequence representation methods itself superiority of TF-IDF weighting over the simple BOW becomes apparent in all configurations.

5 Summary & Discussion

The presented results show that methods from the NLP domain are powerful tools to find meaningful representations of SCADA log-messages and sequences. They allow to capture and encode syntactic and semantic similarities into the respective vector representations. This enables and facilitates the successful application of off-the-shelve ML algorithms for CM-related tasks. The proposed COALS-t method has shown to be particularly useful for analysis where relations between messages play an important role, such as log-message clustering or the investigation of causal relations between messages. The same holds for sequence embeddings based on COALS-t representations. On the other hand, sequences represented by a sequence of one-hot-encoded messages have shown to allow a sharper distinction between sequences related to different error messages. This could be a useful characteristic in an anomaly detection setting. Moreover, the results show that using a TF-IDF instead of a BOW approach amplifies the observations described above and is therefore preferable. However, this study represents only an initial step towards harvesting the full potential of SCADA-log information for data-driven CM applications. Further research has to prove the value of the proposed representations for respective CM-related tasks.

References

[1] Tautz-Weinert J and Watson S J 2016 *IET Renewable Power Generation* **11** 382–394

[2] Leahy K, Gallagher C V, Bruton K, O'Donovan P and O'Sullivan D T 2017 *Journal of Physics: Conference Series* (IOP Publishing) pp 1–14

[3] Chen B, Qiu Y, Feng Y, Tavner P and Song W 2011

[4] Qiu Y, Feng Y, Tavner P, Richardson P, Erdos G and Chen B 2012 *Wind Energy* **15** 951–966

[5] Leahy K, Gallagher C, O'Donovan P and O'Sullivan D T 2018 *IET Renewable Power Generation* **12** 1146–1154

[6] Rohde D L, Gonnerman L M and Plaut D C 2006 *Communications of the ACM* **8** 116

[7] Liu Z, Lin Y and Sun M 2020 *Representation Learning for Natural Language Processing* (Springer Nature)

[8] Maaten L v d and Hinton G 2008 *Journal of machine learning research* **9** 2579–2605