Online Classification of Alarm Floods Using a Word2vec Algorithm

Nicola Tamascelli*‡[1], Harikrishna Rao Mohan Rao†[1], Valerio Cozzani‡[2], Nicola Paltrinieri*[2], Tongwen Chen†[2]
*Department of Mechanical and Industrial Engineering,
Norwegian University of Science and Technology, Trondheim, Norway
Email: [1]nicola.tamascelli@ntnu.no, [2]nicola.paltrinieri@ntnu.no
†Department of Electrical & Computer Engineering,
University of Alberta, Edmonton, Alberta T6G 1H9, Canada
Email: [1]mohanrao@ualberta.ca, [2]tchen@ualberta.ca
‡Department of Civil, Chemical, Environmental, and Materials Engineering,
University of Bologna, Bologna, Italy
Email: [1]nicola.tamascelli2@unibo.it, [2]valerio.cozzani@unibo.it

***Abstract*—Alarm floods are periods of intense alarm activity that may hinder control room operators' ability to diagnose and respond to process abnormalities. In this context, a method to guide and assist operators during alarm floods would provide critical support in preventing abnormalities from escalating into serious accidents. Therefore, this study introduces a novel approach for the online classification of alarm floods based on their fault categories. Historical alarm data are used to train an ensemble of Natural Language Processing models, specifically word2vec, which learn contextual relationships between alarms under different fault conditions. As a new alarm flood appears, the models predict the most probable context alarms by exploiting the knowledge gained during training. Finally, a scoring system is proposed to reward the models that make correct predictions and eventually identify the most probable fault category. The efficacy of the method has been tested on simulated alarm data from the Tennessee Eastman Process benchmark. The results are encouraging, as the models achieved relatively high accuracy in most fault categories.**

***Index Terms*—Alarm Floods, Online Classification, Word2vec.**

I. Introduction

Alarm systems are integral to modern process plants ensuring their safe and efficient operation, necessitated by their increasing complexity and the demanding production requirements [1], [2]. The advances in digital technology have introduced complex monitoring and alerting capabilities, making it convenient to design and configure alarms. However, the ease of adding alarm points has resulted in numerous alarm management problems, including alarm floods - the presence of a large number of alarms beyond what a plant operator can efficiently handle at a time. The industrial standards, ISA [3] and EEMUA [4], define an Alarm Flood (AF) as a period having 10 or more annunciated alarms per 10 minutes per operator and recommend that an operator shall receive no more than 6 alarms/hour.

During AFs, operators may be overwhelmed by the numerous alarms distracting them from addressing critical alarms and ongoing abnormalities, resulting in potentially dangerous situations. AFs have contributed to catastrophic incidents, including the Three Mile Island (1979), Chernobyl disaster (1986), Texaco Refinery (1994), among others. In addition to compromising safety, the presence of AFs can significantly reduce the efficiency and performance of alarm systems. Due to the complex connectivity and interactions, the fault originating at one point can lead to a cascade of alarms. Furthermore, alarm sequences originating from the same fault category are expected to be similar, and analyzing AFs based on the alarms and their sequential order can provide insights into the root causes of the associated abnormalities. However, this task is challenging due to the presence of noise and varying fault conditions, which can lead to mismatches in alarm sequences. Therefore, advanced techniques are needed to effectively analyze AFs through accurate pattern matching and similarity calculation.

Research interest in AF analysis, classification, and prediction has increased over the recent years [5]. Based on the implementation, these methods can be broadly classified into offline and online techniques. Offline techniques identify similar similar AF sequences based on various similarity metrics to provide decision support for operators. Cheng *et al.* modified the Smith-Waterman algorithm to identify similar AF patterns [6]. The computational complexity of the approach in [6] was addressed through a local alignment approach based on the basic local alignment search tool (BLAST) in [7]. The order-ambiguity of alarms in AF sequences was addressed using extended term frequency-inverse document frequency (TF-IDF)-based clustering approaches [8] and a modified PrefixSpan algorithm considering AFs as time-stamped sequences in [9]. Manca *et al.* used dynamic causal dependencies of highly affected process variables for early detection of AFs.

Online alarm flood analysis uses advanced machine learning techniques to identify and classify ongoing alarm floods, enabling early detection of potential root causes of abnormal conditions, and enabling plant operators to take corrective actions before the situation escalates. Various approaches have been

This work was partially supported by the Natural Sciences and Engineering Research Council of Canada.

proposed, such as incremental dynamic programming [10], a binary series classification using Support Vector Machines and k-Nearest Neighbors [11], and Exponentially Attenuated Component Analysis that prioritizes alarms triggering earlier in the alarm flood [12]. Furthermore, operator assistance systems were developed using a Natural Language Processing (NLP) technique, namely, bag-of-words in [13] and real-time pattern matching and alarm ranking approach in [14]. Finally, Wang *et al.* utilized HAZOP analysis to identify abnormal scenarios and built an online model for process monitoring using a Bayesian network of process variables in [15].

Despite the advances in the field of data mining and computational technology, online alarm flood classification remain under-explored in the literature. This can be attributed to the computational complexity of advanced algorithms, which limits their implementation in online settings. Recently, the field of machine learning has made progress in proposing simple and robust Natural Language Processing (NLP) techniques, which are applied to various tasks such as chatbot development, language translation, sentiment analysis, text generation, question answering, and more. For example, the latest release of the GPT (Generative Pre-trained Transformer) series by OpenAI [16], GPT-4 brings a new approach to language models that can provide better results for NLP tasks. Nevertheless, there are still few studies that utilize NLP techniques for the online classification of alarm floods. Motivated by the above problem and the gap in the literature, we propose a novel and computationally efficient approach for the online classification of alarm floods using word2vec, an advanced NLP technique. The main contributions are:

1) The most probable alarms in the ongoing alarm flood are predicted by capturing the contextual relationships between the alarms in different fault conditions.
2) To reduce the computational complexity, a scoring system is utilized to classify the ongoing alarm floods, thereby removing the need for an additional classification or clustering algorithm.

The rest of the paper is organized as follows. Section II presents the detailed steps involved in the online classification of alarm floods. The effectiveness of the proposed method is demonstrated via a case study in Section III, followed by concluding remarks in Section IV.

II. Methodology

Details of the proposed method for the online classification of AFs using the word2vec algorithm are presented in this section, where the approach has two main stages, namely, offline training of the models and online AF predictions.

A framework of the method is provided in Fig. 1, where the steps in offline training of the models are shown in blue and the steps in online AF prediction are shown in green. Specifically, the offline stage involves the preprocessing of alarm data and training of an ensemble of word2vec models using a cluster of similar AFs; whereas, in the online stage, the ongoing AF is analyzed using trained models to predict the most probable

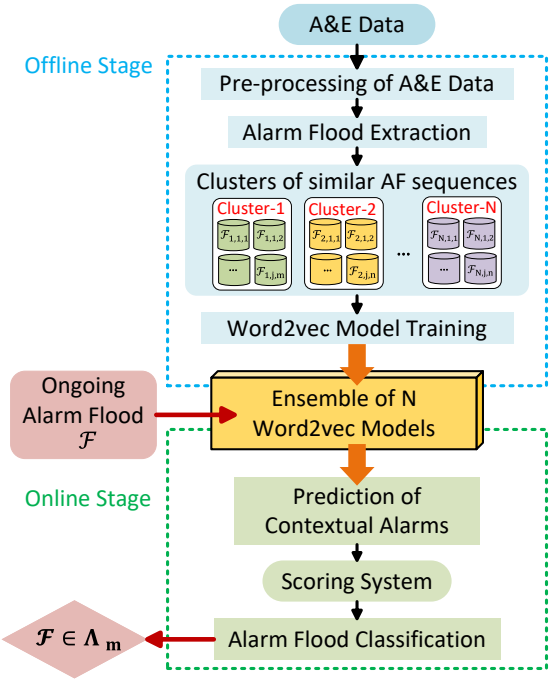


Fig. 1: Framework of the proposed method, consisting of two main stages, namely, offline training of the model (highlighted in blue) and online alarm flood classification (highlighted in green). The algorithm classifies the ongoing alarm flood $\mathcal{F}$ as belonging to a known fault category Λ_m.

alarm and a scoring system is introduced to classify the AF into specific fault categories.

A. *Offline Stage - I: Data Pre-processing & AF Extraction*

In the offline stage, an ensemble of word2vec models is trained using alarm floods sequences extracted from historical Alarm & Event (A&E) data, where the calculations are performed in three steps, including the pre-processing of A&E data, alarm flood extraction, and model training.

1) Pre-processing of A&E Data: To systematically extract the contextual relationships between alarms, the historical data is pre-processed to obtain an ensemble of word2vec models. An A&E log is a chronologically ordered series of alarm events in textual form, where an alarm event is defined as

$$\mathcal{E} = (a, m, t), \tag{1}$$

where $a \in \mathcal{A}$ is the alarm, $m = \{0, 1\}$ is the status of the alarm at time $t \in \mathcal{T}$. Here, $\mathcal{A}$ represents the set of alarms configured in the plant, and $\mathcal{T}$ is the time duration for which the A&E data was collected. Furthermore, an alarm a is characterized by an alarm tag and identifier as $a = (\alpha, \nu)$, where α is the alarm tag, which contains the information about the area and component to which the alarm is configured, and ν provides the details about the type of alarm. For instance, the alarm "PI100.LL" is a combination of the alarm tag PI100 (indicating Pressure Indicator belonging to control loop number 100) and the identifier "LL" (indicating an analog alarm LowLow). Thereafter, chattering alarms are identified

and discarded because such alarms are typically a result of noise or disturbance in the process.

2) Alarm Flood Extraction: An A&E log may contain sequences of alarms due to multiple process faults and disturbances. As mentioned in Section I, the alarm sequences are considered to be similar if they originated from the same fault category. Therefore, labeled clusters of similar alarm sequences, with pre-identified fault categories are taken as the input to the offline stage. Define an alarm sequence from the A&E log as

$$\mathcal{S} = \langle \mathcal{E}_1, \mathcal{E}_2, \cdots, \mathcal{E}_{|\mathcal{S}|} \rangle, \tag{2}$$

where, $\mathcal{E}_k$ represents the kth alarm event, $k = 1, 2, \cdots |\mathcal{S}|$; the operator $\langle \cdot \rangle$ indicates a sequence; and the operator $|\cdot|$ gives the size of the alarm sequence. If the analysis focuses only on the alarms triggered ($m = 1$) and the time of occurrence is not considered, the alarm sequence can be represented as

$$\mathcal{S} = \langle a_1, a_2, \cdots, a_{|\mathcal{S}|} \rangle, \tag{3}$$

where $\mathcal{S}$ is in the form of strings (textual data). Consider the A&E log consists of alarm sequences from N fault categories, $\Lambda = \{\Lambda_1, \Lambda_2, \cdots, \Lambda_N\}$, and it is assumed that there exists at least one alarm sequence in each fault category, Λ_i. The alarm sequences associated with Λ_i can be grouped into a cluster $\mathcal{C}_i$

$$\mathcal{C}_i = \{\mathcal{S}_{i,1}, \mathcal{S}_{i,2}, \cdots, \mathcal{S}_{i,|\mathcal{C}_i|}\}, \tag{4}$$

where, $\mathcal{S}_{i,j}$ represents the jth alarm sequence in the cluster associated with the Λ_i. Here, $|\mathcal{C}_i| \geq 1$ or $\mathcal{C}_i \neq \emptyset$. Finally, the clusters of alarm sequences from the N fault categories are collected into a set as

$$\mathbb{C} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_N\}. \tag{5}$$

Thereafter, AF sequences are extracted from these clusters following the definition in [3]. An AF starts when the alarm rate (namely, the number of alarms within a time window Δt) exceeds an upper threshold τ_{max} and ends when the alarm rate drops below a lower threshold τ_{min}. Therefore, a binary indicator σ is defined to differentiate between alarms that belong (σ=1) and do not belong (σ=0) to a flood. The indicator σ of an alarm event $\mathcal{E}_k$ can be defined as

$$\sigma_{(\mathcal{E}_k)} = \begin{cases} 1, & \text{if } \Gamma \geq \tau_{max}, \\ 1, & \text{if } \Gamma \geq \tau_{min} \text{ and } \sigma_{(\mathcal{E}_{k-1})} = 1, \\ 0, & \text{if } \Gamma < \tau_{min}, \end{cases} \tag{6}$$

where Γ denotes the alarm count within $t_k + \Delta t$. The indicator σ is used to extract AFs from each alarm sequence. As a result, clusters of alarm sequences in (5) are converted into clusters of alarm floods as

$$\mathbb{F}_{i,j} = \{\mathcal{F}_{i,j,1}, \mathcal{F}_{i,j,2}, \cdots, \mathcal{F}_{i,j,|\mathbb{F}_{i,j}|}\}, \tag{7}$$

where, $\mathcal{F}_{i,j,k}$ represents the k^{th} flood extracted from $\mathcal{S}_{i,j}$. It is worth noting that an alarm sequence $\mathcal{S}$ may contain more than one AF depending on process dynamics. Afterward, the AF sequences are represented in the form of a list of strings as in (3), by removing the time stamps associated with the alarms to be compatible with the requirements of the word2vec model.

B. Offline Stage - II: Preliminaries & Model Training

Some preliminaries of the word2vec algorithm and the detailed steps for training the model are provided.

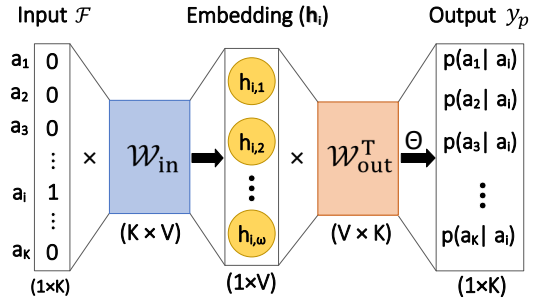


Fig. 2: The framework of Skipgram model, where the input layer is a K dimensional one-hot encoded representation of a_i, the embedding layer $\mathbf{h}_i$ is the vector representation of a_i, and the output layer is the conditional probability $p(a_k|a_i)$, $k = 1, 2, \cdots, K$. $\mathcal{W}_{in}$ and $\mathcal{W}_{out}$ are the internal weights of the model. Θ is the softmax transformation function.

1) Word2vec Algorithm: The proposed work utilizes a widely used NLP model "word2vec", that transforms words in a text into fixed-length vectors (word embeddings) capturing their semantic and syntactic relationships in a high-dimensional vector space [17]. In semantic analysis, this model is used to predict contextual words in textual data, where the context of a word in a sentence is described by the words preceding and succeeding it. In this study, we consider that a cluster of similar AFs is analogous to a collection of topic-specific texts, where alarms are analogous to the words composing sentences. Therefore, the word2vec model is adapted to suit alarm flood applications to predict context alarms, where the context of an alarm in an alarm sequence is defined by the abnormal situation that triggered the alarm (namely, the fault category). Specifically, the context of an alarm a_i in an alarm sequence $\mathcal{S}$ is featured by the alarms occurring within a short temporal vicinity of a_i or in other words, the alarms preceding and succeeding a_i. This study uses the Skipgram architecture of the model to predict context alarms based on an input set of alarms, defined as target alarms [17]. Fig. 2 provides the framework of the Skipgram architecture, where the input to the model is a one-hot encoded representation of an alarm a_i, and the model is a single-layer Neural Network that converts the target alarm a_i into a vector $\mathbf{h}_i$ of cardinality V and generates the output layer of conditional probabilities $p(a_k|a_i)$, $k = 1, 2, \cdots, K$. Here, $K = |\mathcal{A}|$, the number of unique alarms configured in the system. Thereafter, the model is trained to be used for online alarm flood classification.

2) Model Training: Each cluster of AFs is utilized to train a word2vec model as in [17], such that each model learns contextual similarities between alarms based on the fault category associated with the AF. The model is trained using the cluster of AFs to obtain two matrices $\mathcal{W}_{in}, \mathcal{W}_{out} \in \mathbb{R}^{\mathrm{K \times V}}$, representing internal weights. Here, K is the number of unique alarms configured and V is a user-defined parameter representing the cardinality of the word embedding. Each row of $\mathcal{W}_{in}$ contains word embedding of a specific alarm, whereas rows of $\mathcal{W}_{out}$ represent contextual information between alarms.

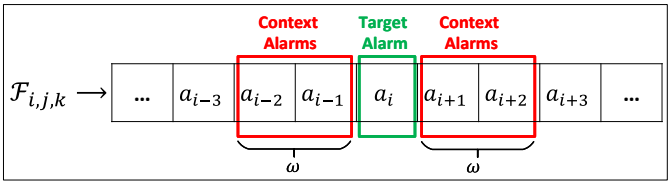


Fig. 3: An example of target and context alarms in a flood. Here, a_i represents the ith alarm, $\mathcal{F}_{i,j,k}$ represents the incoming alarm flood, and ω is the user-defined parameter (window size) to determine the number of context alarms.

Thereafter, $\mathcal{W}_{in}$ and $\mathcal{W}_{out}$ are tuned to learn the contextual relationship between alarms.

One model is trained on each AF cluster, where it iterates over the alarms in each flood sequence. The concept of "context alarm" and "target alarm" is better explained using Fig. 3, where an example of an alarm sequence of an ongoing AF is analyzed. As the model iterates over each alarm in the flood, the alarm currently being analyzed (a_i) is referred to as the "target alarm" and the alarms in the vicinity of a_i, namely, a_{i-2}, a_{i-1}, a_{i+1}, and a_{i+2}, are referred to as the "context alarms". The user-specified parameter $\omega \in \mathbb{N}^+$ (window size) determines the number of "context alarms". The conditional probability of an alarm a_k being a context alarm for the target alarm a_i is obtained from the output y_p as the softmax function transformation $\Theta(\cdot)$ of $\mathbf{h}_i \cdot \mathcal{W}_{out}^T$ given by [18],

$$y_p = \Theta\left(\mathbf{h}_i \cdot \mathcal{W}_{out}^T\right) = \begin{bmatrix} p(a_1 \mid a_i) \\ p(a_2 \mid a_i) \\ \cdots \\ p(a_K \mid a_i) \end{bmatrix}, \tag{8}$$

where, y_p is a vector of dimension K, and satisfies that $\sum_{k=1}^{K} p(a_k|a_i) = 1$. Afterward, the weights $\theta = [\mathcal{W}_{in}, \mathcal{W}_{out}]$ of the model are tuned to minimize the prediction error $(y_p - y_{true})$. Here, y_{true} is a one-hot encoded vector, with the conditional probability of the true context alarm, $p(a_k \mid a_i) = 1$, and of the rest alarms is 0. The internal parameters of the model are tuned to maximize the probability of predicting all the correct context alarms based on

$$\hat{\theta} = \arg\min_{\theta} \left(-log \prod_{c} p(a_c \mid a_i) \right), \tag{9}$$

where $\hat{\theta}$ represents the updated model weights, and c indicates true context alarms. One word2vec model is trained for each cluster $\mathcal{C}_i$ of similar AF, where the model embeds the contextual relationships between alarms in a specific fault, resulting in an ensemble of N models, $\Phi = \{\Phi_1, \Phi_2, \cdots, \Phi_N\}$. Therefore, it can be seen that a cluster $\mathcal{C}_i$ would be associated with a fault Λ_i and a word2vec model Φ_i.

C. Online Stage: AF Prediction and Classification

In the online stage, the alarms in an ongoing AF are fed to the ensemble of word2vec models obtained in the offline stage. Each of the alarms is considered a target alarm and the most probable context alarm is determined as the alarm with the highest conditional probability in y_p. Consider the ongoing AF $\mathcal{F}$,

$$\mathcal{F} = \{a_1, a_2, ..., a_{|\mathcal{F}|}\}, \tag{10}$$

where, a_i represents the ith alarm in $\mathcal{F}$, $i = 1, 2, \cdots, |\mathcal{F}|$. The N ensemble models predict the most probable context alarms based on contextual relationships captured in the offline stage. As the AF proceeds, the predictions are updated as

$$\mathcal{Y}_p = [y_{p,1}, y_{p,2}, \cdots, y_{p,|\mathcal{F}|}], \tag{11}$$

where, $\mathcal{Y}_p$ is the matrix of predictions, obtained by the concatenation of rank-ordered predictions $y_{p,i}$, and $i = 1, 2, \cdots, |\mathcal{F}|$. Specifically, elements in $y_{p,i}$ are rearranged (sorted) in descending order before concatenation. Thus, each model returns a matrix of predictions $\mathcal{Y}_p \in \mathbb{R}^{\mathrm{K} \times |\mathcal{F}|}$, where each column represents the predictions corresponding to alarm $a_i \in \mathcal{F}$. The top n predictions are selected from $\mathcal{Y}_p$ as

$$\overline{\mathcal{Y}}_{p,n} = [\overline{y}_{p,1}, \overline{y}_{p,2}, \cdots, \overline{y}_{p,|\mathcal{F}|}] = [\pi_1, \pi_2, \cdots, \pi_n]^T, \tag{12}$$

where, $\overline{y}_{p,i}$ is the ith column of $\overline{\mathcal{Y}}_{p,n}$, and π_1 (π_n) is the first (nth) row in $\mathcal{Y}_p$, and it satisfies that $\pi_{1,i} \geq \pi_{2,i} \geq \cdots \geq \pi_{K,i}$, where $\pi_{i,j}$ represents the (i,j)th value of $\mathcal{Y}_p$. Thereafter, the alarms corresponding to each prediction are obtained in the form of a matrix as

$$\mathcal{M} = [\mu_1, \mu_2, \cdots, \mu_{|\mathcal{F}|}] = \{a_{i,j} | a_{i,j} \succ \pi_{i,j}, a \in \mathcal{A}\}, \tag{13}$$

where $\mathcal{M}$ is the matrix of alarms, μ_i is the ith column of $\mathcal{M}$, and the operator $\succ$ indicates that the alarm $a_{i,j}$ is corresponding to the prediction value $\pi_{i,j}$. Thus, the output of

Algorithm 1: Online AF Prediction and Classification

Input: $\mathcal{F}$, Λ, Φ, ω_t, n
Output: Λ_{out}

```
S = {s_1 = 0, s_2 = 0, ..., s_|Φ| = 0}   ▷ Initialize scores
for a_i in F do
    Obtain the index i of a_i in F
    for Φ_j in Φ do
        Find the index j of Φ_j in Φ
        Calculate y_p for a_i from Φ_j by (8)
        Obtain ȳ_{p,i} the top n predictions by (12)
        Obtain the corresponding alarms μ_i by (13)
    end
    for a_past in F[i − ω_t : i] do
        if a_past ∈ μ_i then
            s_j = s_j + 1                ▷ Increase the score
        end
        Calculate y_p for a_past from Φ_j by (8)
        Obtain ȳ_{p,past} for a_past by (12)
        Obtain μ_past by (13)
        if a_i in μ_past then
            s_j = s_j + 1                ▷ Increase the score
        end
    end
end
Find the index k of the highest score in S
The fault category Λ_out of F is the kth element of Λ
return Λ_out
```

the model can be interpreted as a list of alarms rank-ordered based on their likelihood of occurrence as a context alarm. In other words, the first column of $\overline{\mathcal{Y}}_{p,n}$ represents the n most probable context alarms of $a_1 \in \mathcal{F}$.

Furthermore, an incremental scoring system is introduced to reward models for generating correct predictions. At the beginning of the predictions, the scores are initialized to 0 and are incremented by one unit for each correct prediction. The algorithm requires two user-specified parameters, namely, ω_t and n. Here, $\omega_t \in \mathbb{N}^+$ determines the number of context alarms to be considered in the ongoing AF sequence. It has to be noted that ω_t has the same purpose as that of ω utilized in the model training, as described in Section II-B, and it is not necessary that $\omega_t = \omega$. The parameter $n \in \mathbb{N}^+$ determines the number of top predictions to be selected in (12) to identify the most similar AF sequence.

If an AF is triggered due to a fault Λ_i, the model $\Phi_i \in \Phi$ trained on the cluster $\mathcal{C}_i$ corresponding to Λ_i would give the most accurate predictions and hence would result in the highest number of similar context alarms. This principle is utilized to classify the ongoing AF, i.e., the AF is classified into the fault category of the model with the highest score. Algorithm 1 summarizes the online AF Prediction and Classification.

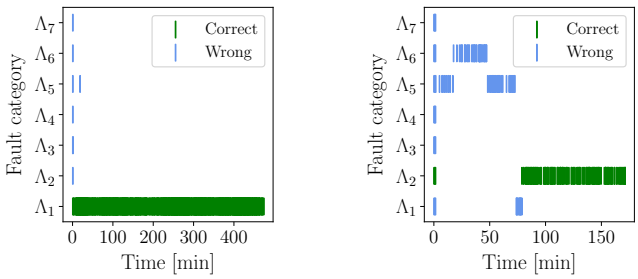
III. Case Study

The applicability and effectiveness of the proposed method are demonstrated through a case study using simulated alarm data from the benchmark Tennessee Eastman Process (TEP).

A. Description of the Simulated Data

The closed-loop simulator of the benchmark Tennessee Eastman Process, developed by Bathelt *et al*. [19] was utilized in this study. The alarm data was prepared following the procedure in [12]. Specifically, seven faults $\Lambda = \{\Lambda_1, \Lambda_2, \cdots, \Lambda_7\}$ were simulated by introducing disturbances as step inputs or valve stiction. Four types of alarms were configured on 52 process variables (PV), namely, "PV.HH" (High-High), "PV.H" (High), "PV.L" (Low), and "PV.LL" (Low-Low), resulting in 208 unique alarms. For each fault category, a set of 40 independent simulations were performed, where the duration of each simulation was 10h and the faults were introduced after 2h of steady-state operation. The chattering alarms were identified and discarded. Alarm floods were extracted using (6), and 7 clusters with 40 AF sequences each were obtained. Here, the alarm sequences generated by a specific fault were regarded as a cluster of similar alarm sequences. The number of AF sequences in each fault category is as follows: $|\mathcal{C}_1| = 41$, $|\mathcal{C}_2| = 71$, $|\mathcal{C}_3| = 25$, $|\mathcal{C}_4| = 1$, $|\mathcal{C}_5| = 42$, $|\mathcal{C}_6| = 66$, and $|\mathcal{C}_7| = 40$. Due to insufficient AF sequences, cluster $\mathcal{C}_4$ associated with fault Λ_4 was excluded from the analysis.

Thereafter, an ensemble of 6 word2vec models was trained using a Leave-one-out cross-validation approach. Specifically, the first 20 flood sequences of each fault category were selected to evaluate the performance of the models. Therefore, 20 independent simulations have been performed such that in each simulation, models were trained on all the AF sequences except one (test sequence) and the last AF sequence was used to evaluate the model performance and tuning. Subsequently, the online alarm flood classification is performed using this ensemble of word2vec models.



a) True category → Λ_1 b) True category → Λ_2

Fig. 4: Online classification of two floods that originated from fault category (a) Λ_1 and (b) Λ_2. The vertical lines represent the result of the classification, where the correct (incorrect) predictions are shown in green (blue).

B. Results and Discussion

The models were trained using the open-source Python library Gensim [20] v4.2.0 running on Python v3.9.12. For the offline stage, the parameters used were $\omega = 5$, $V = 15$, and the number of epochs = $100,000$, which indicates the number of iterations made by the model over the dataset. For the online predictions and classification, the parameters used were $\omega_t = 3$ and $n = 10$. Fig. 4 provides an example of the output from the online prediction and classification stage. The vertical lines represent the result of the classification, where the correct (incorrect) predictions are shown in green (blue). Fig. 4(a) shows the accurate prediction and classification of the AF sequence to be associated with fault Λ_1, without any significant delay. However, the AF sequence in Fig. 4(b) was classified into fault Λ_2 after about 70 minutes.

To provide a comprehensive overview of the model performance, the class-wise accuracy has been calculated by considering all predictions obtained for each fault category. In addition, to evaluate if the model was able to accurately classify the AF sequence using the complete AF sequence, the class-wise accuracy was determined by considering only the last prediction (i.e., the last alarm of a flood sequence). These performance metrics of the model using class-wise accuracy are summarized in Table I. It can be seen that the models achieve prediction accuracy above 0.85 (all predictions) and 0.90 (last predictions), respectively, in the identification of fault categories Λ_1, Λ_3, and Λ_5. The model performance is satisfactory for category Λ_6, which shows an accuracy of 0.77 considering all the predictions. However, the identification of fault category Λ_2 is particularly challenging (accuracy = 0.37).

Furthermore, the results indicate that the model performance increases as the flood proceeds because the accuracy based on the last prediction is always greater than the accuracy based on all predictions. This behavior is especially evident for categories Λ_2 and Λ_3 and it may indicate that most errors are made during the early stages of AFs (see Fig. 4.b). This

TABLE I: Class-wise accuracy

Accuracy Type	Fault Categories						Mean Accuracy
	Λ_1	Λ_2	Λ_3	Λ_5	Λ_6	Λ_7	
All predictions	0.98	0.37	0.89	1.0	0.77	0.001	0.67
Last prediction	1.0	0.55	0.9	1.0	0.85	0.00	0.72

could be explained by the lack of sufficient information at the beginning of the AF. Finally, it is to be noted that the models could not classify the sequences associated with the fault Λ_7 (accuracy=0.01), and those sequences (belonging to Λ_7) were always incorrectly classified into the fault Λ_2. Further investigation using process knowledge is recommended to identify the reason for such a performance with fault Λ_7.

In summary, the performance of the models is satisfactory, which indicates that the proposed method can support the operator in the real-time monitoring of AFs. The challenges with lower accuracy values for two fault categories could be attributed to the fact that model hyper-parameters were chosen based on the best practices and were not tuned for this specific process or application. The model performance is expected to improve from an exhaustive hyper-parameter tuning using a grid-search algorithm. Further research is recommended to improve the detection performance during the early stage of the AF, employing different NLP algorithms, such as BERT [21] and XLNET [22].

IV. Conclusions

This study presents a novel approach for online alarm flood classification using the word2vec algorithm and historical A&E data. The method articulates in two phases, namely, offline training and online predictions. In the offline stage, the contextual relationships between alarms are captured to train an ensemble of word2vec models using clusters of labeled AF sequences. In the online stage, the most probable context alarms are predicted using the trained ensemble of models, and the AF is classified into appropriate fault categories using a scoring system. Unlike other methods in AF classification, this study utilizes the NLP algorithms not only to learn hidden relationships between alarms but also to perform the classification of ongoing AFs without any classifiers or clustering algorithms, thereby reducing the computational complexity. The approach has been tested on simulated alarm data obtained from the benchmark TEP. The models achieved accuracy in the range of 0.77 to 1.0 in four out of six categories. Additionally, the results indicate that the model performance improves as the AF proceeds, leading to more accurate predictions as more information is available.

Further research is recommended to optimize the model parameters and improve the accuracy during the early stage of AFs. Additional investigation using process knowledge is required in the two fault categories resulting in poor prediction accuracy. Notwithstanding these limitations, the approach shows the potential of NLP algorithms in alarm flood analysis and makes a significant contribution to the novel line of research using NLP models for online AF classification.

References

[1] G. Manca and A. Fay, "Detection of historical alarm subsequences using alarm events and a coactivation constraint," *IEEE Access*, vol. 9, pp. 46 851–46 873, 2021.

[2] J. Wang, F. Yang, T. Chen, and S. L. Shah, "An overview of industrial alarm systems: Main causes for alarm overloading, research status, and open problems," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 1045–1061, 2016.

[3] *ANSI/ISA-18.2: Management of Alarm Systems for the Process Industries*, ISA (International Society of Automation), Durham, NC USA, 2016.

[4] *Alarm Systems: A Guide to Design, Management and Procurement*, EEMUA (Engineering Equipment and Materials Users' Association), London, 2013.

[5] G. Dorgo, F. Tandari, T. Szabó, A. Palazoglu, and J. Abonyi, "Quality vs. quantity of alarm messages-how to measure the performance of an alarm system," *Chemical Engineering Research and Design*, vol. 173, pp. 63–80, 2021.

[6] Y. Cheng, I. Izadi, and T. Chen, "Pattern matching of alarm flood sequences by a modified Smith–Waterman algorithm," *Chemical Engineering Research and Design*, vol. 91, no. 6, pp. 1085–1094, 2013.

[7] W. Hu, J. Wang, and T. Chen, "A local alignment approach to similarity analysis of industrial alarm flood sequences," *Control Engineering Practice*, vol. 55, pp. 13–25, 2016.

[8] G. Manca, M. Dix, and A. Fay, "Clustering of similar historical alarm subsequences in industrial control systems using alarm series and characteristic coactivations," *IEEE Access*, vol. 9, pp. 154 965–154 974, 2021.

[9] Q.-X. Zhu, C. Jin, Y.-L. He, and Y. Xu, "Pattern mining of alarm flood sequences using an improved prefixspan algorithm with tolerance to short-term order ambiguity," *Industrial & Engineering Chemistry Research*, vol. 60, no. 11, pp. 4375–4384, 2021.

[10] S. Lai, F. Yang, and T. Chen, "Online pattern matching and prediction of incoming alarm floods," *Journal of Process Control*, vol. 56, pp. 69–78, 2017.

[11] M. Lucke, M. Chioua, C. Grimholt, M. Hollender, and N. F. Thornhill, "Advances in alarm data analysis with a practical application to online alarm flood classification," *Journal of Process Control*, vol. 79, pp. 56–71, 2019.

[12] J. Shang and T. Chen, "Early classification of alarm floods via exponentially attenuated component analysis," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 10, pp. 8702–8712, 2019.

[13] H. S. Alinezhad, J. Shang, and T. Chen, "Early classification of industrial alarm floods based on semisupervised learning," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1845–1853, 2021.

[14] M. R. Parvez, W. Hu, and T. Chen, "Real-time pattern matching and ranking for early prediction of industrial alarm floods," *Control Engineering Practice*, vol. 120, p. 105004, 2022.

[15] H. Wang, F. Khan, and S. Ahmed, "Design of scenario-based early warning system for process operations," *Industrial & Engineering Chemistry Research*, vol. 54, no. 33, pp. 8255–8265, 2015.

[16] OpenAI, "Gpt-4 technical report," 2023.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations - Workshop Track Proceedings*, 2013.

[18] J. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," *Advances in Neural Information Processing Systems*, vol. 2, 1989.

[19] A. Bathelt, N. L. Ricker, and M. Jelali, "Revision of the Tennessee Eastman Process model," *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 309–314, 2015.

[20] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, 2010.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 2019, pp. 4171 – 4186.

[22] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," *Advances in Neural Information Processing Systems*, vol. 32, 2019.