A Custom Word Embedding Model for Clustering of Maintenance Records

Abhijeet Sandeep Bhardwaj[1], Akash Deep[2], Dharmaraj Veeramani[3] and Shiyu Zhou[4]

***Abstract*—Maintenance records of industrial equipment contain rich descriptive information in free-text format, such as, involved parts, failure mechanisms, operating conditions, etc. Our objective is to leverage this unstructured textual information to identify groups of similar maintenance jobs. We use a natural language based approach and propose a novel custom word embedding model which utilizes two sources of information 1) maintenance records collected from in-field operations and 2) industrial taxonomy, to effectively identify clusters. The advantages of our model include (a) combined use of semantic and taxonomic sources of information for clustering, (b) one step/simultaneous training, which enables knowledge sharing between the two information sources and reduces hyperparameters, and (c) no dependency on third-party data. We demonstrate the efficacy of our model for cluster identification using a real-world dataset. The results show that simultaneous incorporation of semantic and taxonomic information enables accurate extraction of contextual insights for improving maintenance decision-making and equipment reliability.**

***Index Terms*—Clustering, Natural Language Processing, Maintenance records, Taxonomy.**

I. Introduction

THIS research is motivated by the unstructured free-text information that is documented by technicians when they perform maintenance actions on industrial equipment. These maintenance records contain rich information (related to equipment components, their condition and failure mechanism) that can aid maintenance technicians with fault prognosis [1], root-cause analysis [2], [3] and maintenance decision-making [4], [5]. For the equipment manufacturer, insights from these maintenance records can help in improving equipment reliability through design changes, and thereby reducing the warranty costs, thus gaining a competitive advantage [6].

While it is common practice in industry to create and store maintenance records, it is impractical to manually review and extract insights from the large quantity and variety of records that are typically available [7]. Automatically extracting useful insights from the maintenance records is, however, not trivial. Consider, for instance, the maintenance records from a company having multiple oil rigs. This dataset comprises maintenance records from a variety of oil rig equipment, each having several sub-units with associated maintainable items, and ultimately the specific parts that underwent repair or replacement. For each maintenance action, a record is available which contains structured as well as unstructured information. Structured information consists of well-defined data such as time of action, type of action (corrective or preventive), rig number etc., whereas, unstructured information is the textual description entered and updated by the technicians such as equipment condition, explanation of the wear or failure, maintenance actions performed, components replaced, etc. Fig. 1 shows sample maintenance records for mudpump equipment and includes information specific to the sub-units (i.e., pump and fluid end), maintainable items (e.g., discharge module, suction module) and parts (e.g., valve, seat, oil pressure switch) that were involved in these maintenance actions. While such information are a rich source of insights, they cannot be recorded as structured data.

The aim of this research is to create models to analyze maintenance records and automatically extract groups or clusters of records that are similar (e.g., in terms of the failure mechanism or the part that was repaired or replaced). From a practical perspective, our methodology for analyzing and clustering textual data by combining contextual information as well as industry-specific taxonomic information, will assist industrial equipment manufacturers and operators to enhance their ability to perform analysis of maintenance activities, failure types, and equipment components that were impacted. Using this method, we can extract structured information from unstructured data and then conduct quantitative modeling [8] and analysis of system reliability. For example, using this method, from maintenance record, we can create the failure event history for a specific component failing due to a specific failure mechanism. Such history is currently obtained manually by an operator processing the natural language maintenance record. However, there are certain significant challenges in determining clusters based on the analysis of the textual information in the maintenance records. First, the records in the dataset may comprise different types of maintenance information that are kept by different personnel on the oil rig (such as equipment downtime reports created by machine operators, maintenance reports created by technicians, parts reports created by purchasing and inventory personnel) [9]. Second, the descriptions and language used by different personnel can differ even when they are referring to the same maintenance event. For example, in Fig. 1, it can be seen that phrases like ‘wash out’ or ‘worn’ are used interchangeably by industry personnel when referring to excessive wear of fluid end components. Third, the implied meaning of certain words within the maintenance context can be different from that in general use as discussed in [10]. For example, the word ‘stick’ within the maintenance context most likely implies cohesion as opposed to a piece of wood, thus requiring us to consider the context in which the word is being used. Fourth, the manner in which the clustering of the maintenance records needs to be done also depends upon the desired context and basis for the grouping (e.g., failure

1, 2, 3, 4: Department of Industrial & Systems Engineering, University of Wisconsin-Madsion, Madison, Wisconsin 53706, USA

Input Data			Desired Output	
Job ID	**Rig ID**	**Description of Maintenance Action**	**Failure Mechanism**	**Associated Parts**
J01	R01	Mud pump is currently being run for drilling operations. **Unusual vibrations** noted. **Main Bearing** temp 46c.Mechanics to check temperature when running &also check for vibration	Vibration-Mechanical failure	Bearing-Power End
J02	R01	Parts Requested from ICS:, Mud Pump 2 **Lube Oil Pressure Switch Sticking**. Order new switch, Part received and installed	Sticking-Mechanical failure	Pressure Switch-Lubrication System
J03	R02	Equipment is out of Service Mud pump # 1out of service to carryout change out of **washed-out discharge module**	Washed-Material Failure	Discharge Module
J04	R03	**Valve and seat on suction module** cyl #3were changed out due to **worn conditions** and high hours parts. Parts Requested from ICS:	Worn-Material Failure	Valve & Seat-Suction Module

Fig. 1: Sample of input data for text analysis.

mechanism or parts affected). As a result, our model needs to be flexible to accommodate these user-defined preferences.

Within the reliability literature, efforts have been made by researchers to incorporate textual knowledge. In [11] author propose a graphical semi-supervised industry-specific word embedding approach for classifying documents. There exists a large body of methods for unsupervised text clustering within the natural language processing (NLP) domain [12]. However, for clustering methods, the numerical representation of the textual data is highly imperative. Word2vec has proven to be very effective for generating numerical representation of textual data [13]. The idea is to efficiently represent a single word into a one dimensional vector space (also called embeddings) and then use this representation to perform downstream tasks, like clustering. In general use, word2vec is employed on a single set of data, i.e., the representation generated using word2vec only learns the semantic information presented by the contextual words in the document. This limits the introduction of any kind of external context. Naturally, researchers have provided several ways to tackle this shortcoming. For example, [14] propose a two-step procedure to identify the relations between words, wherein, in the first step they learn embedding, and in the second step, they introduce relation in a supervised way. [15] incorporate the relationships between pairs of words by optimizing different cost functions. However, for these methods, it is unclear how commonality on a sentence or document level can be identified.

[16] use an adaptive clustering module that cluster topics into sub-topics and sample documents relevant to the sub-topics to learn local embeddings. However, their method relies on sampling documents from the internet that are specific to sub-topics present in the taxonomy. In [17], the authors develop a dynamic weighting neural network and use the hypernym-hyponym (parent-child) pairs available in *Wordnet* (a lexical database of semantic relations between words) along with contextual words present in a document. However, the keywords indicating taxonomic/external knowledge may not co-occur together in the same document as can be seen in the illustration of Fig. 1 (where 'vibration' and 'sticking' do not occur together). The dynamic distance margin model proposed by [14], also uses co-occurring words and their frequency to learn their embeddings. Their loss function minimizes the distance between embeddings of the words that exhibits one of the three relations (co-hypernym, co-hyponym and hypernym-hyponym) and thus neglects semantic/contextual information. The Attract-Repel model proposed by [18] uses a pre-specified word representation/embedding and incorporates the external/taxonomic information over it. The word embeddings generated rely on a pre-trained word representation and incorporate taxonomic knowledge as a post processing step [19] This also increases the number of hyper-parameters, like regularization constants, to retain the semantic information. The model proposed by [19] and [20] learn the word embeddings using the corpus and taxonomy information where the contextual knowledge is learnt using the GloVe loss function [21]. However, their model incorporates the taxonomic guidance for only those words which are contextual and co-occur with each other in a given maintenance log.

Recently, transformer based models have become popular in the NLP literature [22], [23]. However, training them require a large corpus like Wikipedia (≈16 GB of uncompressed data) to efficiently learn millions of parameters [24]. Also, such models provide contextual embeddings for words which needs to be processed before applying to a down stream task like clustering [12]. Apart from this, some transformer models which incorporate additional taxonomic information train two tasks simultaneously: 1) Masked Language Modeling and 2) Next sentence prediction tasks. To replicate them, the data has to be transformed in a sequential way (premis-hypothesis for [25] or question-answer as in [26]). Other transformer models like [27] require generation of graph based embeddings while model proposed in [28] require an entity candidate selector. The K-adapter model proposed in [29] require additional supervised dataset to induce the taxonomic information.

A supervised method is proposed in [30], where information about hierarchical taxonomy is incorporated by leveraging local and global classifiers trained on annotation for each word in a record. Similarly, in [31], each word in the sentence is annotated and the objective is to predict not only the contexual words but also the corresponding labels while learning the embeddings. Development of tools to assist annotation of maintenance documents is still under progress [32]. The Dict2Vec model presented in [33] learns embedding for various words using definitions provided in dictionaries like Cambridge, Oxford, etc.,. More recently, the work proposed by [34] learns embeddings for taxonomy terms using their definitions in Wiktionary while the work proposed in [35] aims to classify scholarly articles by learning embeddings for each term in knowledge graph using [36].

Based on the literature review, we note that there is a lack of studies which leverage one-step simultaneous learning of information from both industrial equipment maintenance records and industrial domain taxonomy in a completely unsupervised manner without depending on any additional resource apart from maintenance logs and industrial taxonomy. Hence, in this paper, we propose a novel approach (namely Custom Word Embedding Model (CWEM), which is summarized below:

- We combine Skip-Gram model with standard industrial taxonomy to jointly leverage the information from two sources while learning word embeddings. This helps us to reduce the training time when compared to a two-step

procedure which incorporate the additional information as a post-processing step.
- We employ a one-step learning procedure by employing a new learning parameter which weighs the features to learn from both bodies of information (contextual and taxonomic). This reduces the number of hyper parameters required to tune while replicating the model as opposed to the two-step procedures [18] and thus avoids the dependency on hyper-parameter tuning algorithms.
- The model does not require the taxonomy terms to be contextually co-occurring in a given maintenance log and hence can learn from the complete taxonomy rather than a subset of terms which co-occur.
- The model is learnt in a completely unsupervised manner and avoids dependency on any additional resources like WordNet, Wikipedia or any other supervision.

II. Data Description and Textual Clustering

A. Description of Data Sources

We consider two sources of information. The first is the dataset of maintenance records see Fig. 1. Let L denote the number of maintenance records, each denoted by JobID in the dataset. Corresponding to each JobID we have the associated system and the description of the maintenance action which was conducted as well as information regarding the portion of the equipment that caused the maintenance action, as well as observations regarding the equipment condition. The second source of information which we incorporate is the standard industry taxonomy associated with the oil and gas industry. In particular, we consider two kinds of industrial taxonomies - failure taxonomy and equipment taxonomy. The failure taxonomy provides a list of commonly used technical terms associated with failures and failure mechanisms in oil rig systems. The equipment taxonomy provides the listing of different items that comprise the equipment hierarchy and the relationships between them, namely system, sub-unit, maintainable item and component. In the remainder of this paper we refer to these distinct taxonomy categorizations as classes. A detailed description of the taxonomies is provided in Section IV. Now, suppose the records can be grouped into K different clusters (where $K < L$). Thus, our objective here is to appropriately represent the two information sources and identify the clusters based on a similarity measure. We discuss below the basic steps used in clustering of the textual records.

B. Basics Steps for Clustering of Textual Data

We first introduce key terminology and definitions:

- **Word:** A word is the most fundamental unit used in textual descriptions e.g., “repair”, “leak”, “part”. A word is also referred to as a ‘token’ in this work.
- **Vocabulary:** The set of all available words constitutes a vocabulary. Let $\mathbf{V}$ denote the vocabulary set.
- **Document:** A document is a collection of words which is used to describe any event. For example, each description (maintenance record) in Fig. 1 constitutes a document.
- **Corpus:** The collection of all available documents.

The first step in clustering of textual data is to design an appropriate mathematical representation of these documents. In other words, this mathematical representation is critical in the model’s clustering performance. In the literature, several different representations are available like Latent Semantic Analysis [37], Latent Dirichlet Allocation [38] along with word2vec/Skip-Gram. The state-of-the-art Skip-Gram (SG) model is chosen as the basic representation for our study (we provide more details regarding the SG model in Section III-A). Once we have established a mathematical representation, the next step is to identify the clusters present in the corpus. Several clustering algorithms are available in the literature (please see [39] for a recent review). The core idea of clustering algorithms is to minimize the distance of word representations with respect to cluster centers. We use k-means algorithm for the same which uses a pair-wise distance matrix to identify k-centers of each clusters. We revisit the steps used for clustering in Section IV-C.

III. Model Development

Fig. 2 outlines the workflow of our proposed framework. The central idea of our approach is that it entails learning information from two information sources simultaneously, namely semantic/contextual information and taxonomical information. In the current work, information from a single hierarchy taxonomy, pertaining to the oil and gas industry is incorporated along with the semantic knowledge to learn industry-tailored word embeddings. The industry taxonomy provides a grouping of tokens into different classes where tokens in the same class are similar to each other while tokens in different classes are dissimilar to each other. To incorporate the semantic information, the architecture for the SG model (Section III-A) is used. CWEM, thus, tries to incorporate additional taxonomic information by modifying the loss of the SG model as detailed in Section III-B.

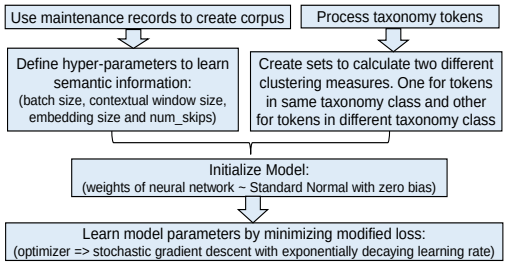


Fig. 2: A flowchart illustrating the steps involved for proposed framework.

A. Distributed Representation of a Word

The semantic information is incorporated using the corpus of maintenance records. We use Skip Gram (SG) [13] model for this purpose (please see [40] for a comprehensive explanation). The central idea is credited to the distributional hypothesis which implies that words which represent similar meaning are generally used in the same context [41]. Fig. 3 provides the basic neural network architecture of the SG model.

The SG model learns a vector representation (known as word embeddings) for an input word by maximizing its affinity with the contextual or neighboring words. For example, consider the sentence “Damage was caused by leaking pump valve”. Here, if we consider the word “leaking” to be the input word, then the rest of the neighboring words are contextual words for it. The number of neighboring words that are used to

learn the vector representation for a given input word constitutes the size of the semantic window represented by $|C|$. The objective of the SG model is to generate a vector representation for the given input word which helps to predict the correct context word with a high accuracy. In the same example, if the size of the semantic window is $|C| = 2$, then the (input, context) word pairs become: (leaking, caused), (leaking, by), (leaking, pump), (leaking, valve). The SG model then learns such embeddings for "leaking" which can give a high output score for the true contextual word. The output score is a soft-max score (equation 1) which assigns a probability ranking to all words in the vocabulary for being observed as the context word for a given input word.

We denote an input word by w_I and its vector representation by $\mathbf{v_{w_I}}$ in the SG model. Word embeddings are learned using a neural network architecture having an *input layer*, a *hidden layer*, and an *output layer*. The input layer is a $|V|$-dimensional layer corresponding to the dictionary created by all V-distinct words present in the vocabulary $\mathbf{V}$ (as shown in Fig. 3). The input layer can thus be considered to have $|V|$ nodes where each node represents a placeholder for every input word. Next, the input layer is connected to the hidden layer $\mathbf{h}$ by a weight matrix $W_{|V|\times N}$. Please note that each row of the weight matrix constitutes the embedding vector $\mathbf{v_{w_I}}$ for the corresponding input word which we are learning. Next, the hidden layer is connected to the output layer using a different weight matrix $W^{'}_{N\times|V|}$ (Please note the $^{'}$ notation is used to refer to elements of the output layers). In the SG model, instead of a single output layer, we have $c \in \mathbf{C} = \{1, 2, 3, \cdots, |C|\}$ panels for the output layer. The weights matrix $W^{'}_{N\times|V|}$ connecting the hidden layer to the output layer is shared among all the panels. The rows of the weight matrix linking the hidden layer to the output layer gives the output vector representation $\mathbf{v^{'}_{w_j}}$ of each word. The final output layer thus has $|C| \times |V|$ nodes, where the c^{th} contextual word located at j^{th} position in vocabulary is denoted by $w_{c,j}$ and its output vector representation is denoted by $\mathbf{v^{'}_{w_j}}$. As the weight matrix $W^{'}_{N\times|V|}$ is shared among all $\mathbf{C}$ panels so is the output vector $\mathbf{v^{'}_{w_j}}$ for a given context word.

The steps followed while training are as follows: First, the hidden layer transposes the input weight vector $\mathbf{v_{w_I}}$ to give hidden layer vector $\mathbf{h} = \mathbf{v_{w_I}}^T$. Second, dot product is evaluated between the hidden layer vector $\mathbf{h}$ and the output weight vector $\mathbf{v^{'}_{w_j}}$ for all words $j \in \mathbf{V}$ at each panel $c \in \mathbf{C}$. The dot-products are passed to a soft-max activation layer which gives us the probability of observing a contextual word $w_{c,j}$ for a given input word w_I at the c^{th} context position. The soft-max score is highest for the true contextual word which are passed while training the neural network as compared to other words in the vocabulary. Following the above example, the word 'pump' should have the largest soft-max score corresponding to the word 'leaking' at the $c = 3^{rd}$ panel (because 'pump' is the 3^{rd} context word for 'leaking').

We now formalize the above steps into mathematical equations. The semantic information for each context word $w_{c,j}$ corresponding to the given input word w_I is learned as a multinomial distribution at each panel $c = \{1, 2, 3, \cdots, |C|\}$ and is given by equation 1. Here, $w_{O,c}$ is the actual c^{th} context word specified while training the neural network. For example, words like 'caused', 'by', 'pump' and 'valve' would correspond to $w_{O,1}$, $w_{O,2}$, $w_{O,3}$ and $w_{O,4}$ for the input word 'leaking'. As already described, for the SG model on the output layer, instead of outputting one multinomial distribution, we are outputting $|C|$ multinomial distributions, one distribution at each panel, where each multinomial distribution gives the probability of observing the true context word at that position. The symbol $u_{c,j}$ represents the net dot-product of the j^{th} word in c^{th} panel with the hidden layer vector $\mathbf{h}$ and is given by $u_{c,j} = u_j = \mathbf{v^{'}_{w_j}}^T \cdot \mathbf{h}$ for $c \in \mathbf{C} = \{1, 2, 3, \cdots, |C|\}$. The cross-entropy loss maximizes the probability of the observed context words for the given input word (equation 2).

$$\textit{soft-max score} = p(w_{c,j} = w_{O,c}|w_I) = \frac{\exp(u_{c,j_{c^*}})}{\sum_{j^{'}=1}^{|V|}\exp(u_{j^{'}})} \tag{1}$$

$$\begin{aligned} &Cross - Entropy - Loss = \sigma(\mathbf{v^{'}_{w_{j^*}}}^T \cdot \mathbf{h}) \\ &= -\log(p(w_{O,1}, w_{O,2}, \cdots, w_{O,|C|}|w_I)) \\ &= -\log(\textstyle\prod_{c=1}^{|C|}\frac{\exp(u_{c,j_c^*})}{\sum_{j^{'}=1}^{|V|}\exp(u_j^{'})}) \\ &= -\textstyle\sum_{c=1}^{|C|}\exp(u_{j_{c^*}}) + |C| \times \log\sum_{j^{'}=1}^{|V|}\exp(u_j^{'}) \end{aligned} \tag{2}$$

where j_c^* is the index of the actual output context word occurring at j^{th} index of the c^{th} panel or occurring at the c^{th} position of the semantic window.

The cross-entropy loss updates the output vector $\mathbf{v^{'}_w}$ for each word w_j in the vocabulary at every iteration and is, therefore, intractable to implement in its original form. To tackle this challenge, [13] proposed the idea of negative sampling where negative samples are drawn along with the true/positive context words. Negative samples (present in set $\mathbf{W_{neg}}$) are random words drawn from the vocabulary that have the least probability to occur as contextual words for a given input word. The Noise Contrastive Estimation (NCE) loss is thus given by equation 3 and minimizes the probability of predicting the negative context word given an input word.

$$\textit{NCE loss} = -\log(\sigma(\mathbf{v^{'}_{w_{j^*}}}^T \cdot \mathbf{h}) - \sum_{w_n \in \mathbf{W_{neg}}}\log(\sigma(-\mathbf{v^{'}_{w_n}}^T\mathbf{h})) \tag{3}$$

where w_{j^*} is the output word (positive sample), $\mathbf{v^{'}_{w_{j^*}}}$ is its output vector, $\mathbf{h}$ is the hidden layer vector $\mathbf{h} = \mathbf{v_{w_I}}^T$, σ is the soft-max activation function and w_n are the negative sampled words having $\mathbf{v^{'}_{w_n}}$ as its vector representation.

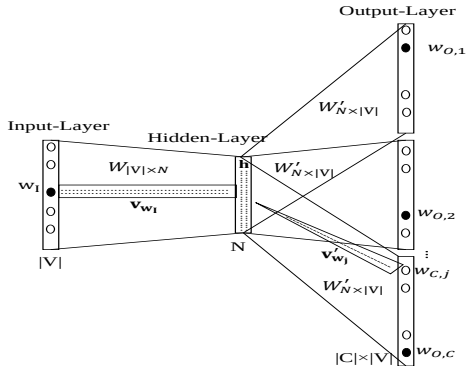


Fig. 3: Neural Network architecture for the Skip Gram model.

B. Proposed Custom Word Embedding Model (CWEM)

Having learned the contextual/semantic information, we now harness the information from the second source i.e., the

taxonomic information. Let $\mathbf{M}$ be the set of classes present in the taxonomy indexed by {'0', '1', '2', $\cdots$, 'm-1'}. For example, 'mechanical failure' and 'material failure' would constitute different classes of the failure mechanism taxonomy. In Fig. 1, the words 'vibration' and 'stick' belong to the same failure mechanism of 'mechanical failure' while the words 'wash out' and 'worn' belong to 'material failure'. The tokens present in the taxonomy are divided into two different sets as shown in Fig. 4. The first set contains pairs of tokens belonging to the same class while the second set contains pairs of tokens belonging to the different classes. The major motivation is that the tokens of the same class should have similar word embeddings while the tokens belonging to different class should have dissimilar word embeddings. Next we define the kinds of similarity measure between the words present in different taxonomic classes which would help include taxonomic information in the learnt embedding.

We define two different kinds of clustering measures. The first measure of *Within Class Dissimilarity* (WCD) tries to move points of the same cluster closer to each other while the second measure of *Between Class Similarity* (BCS) tries to move points of different clusters away from each other. WCD measures the dissimilarity between tokens (words) within the same taxonomy class (as indicated for tokens of taxonomy class 1 in Fig. 4). Intuitively we would like to have the dissimilarity between the embedding vectors of tokens (words) of the same class to be as low as possible. The measure of dissimilarity is given by the sum of the cosine distance between different tokens belonging to the same taxonomy class and is shown in equation (4). Let m_k represent a class in set $\mathbf{M}$. Let w_{T,m_k} define the token (word) from taxonomy class m_k (here the subscript T indicates that the token is also present in the taxonomy). Here, ($\mathbf{v_{w_{T,m_{k_p}}}}$, $\mathbf{v_{w_{T,m_{k_q}}}}$) denote the word vector representation for words $w_{T,m_{k_p}}$ and $w_{T,m_{k_q}}$ belonging to the same class $m_k \in \mathbf{M}$.

$$WCD = \sum_{\forall m_k \in M} \sum_{\substack{w_{T,m_{k_p}} \in m_k \\ w_{T,m_{k_q}} \in m_k}} (1 - \frac{\mathbf{v_{w_{T,m_{k_p}}}} \cdot \mathbf{v_{w_{T,m_{k_q}}}}}{||v_{w_{T,m_{k_p}}}|| \, ||v_{w_{T,m_{k_q}}}||}) \quad (4)$$

BCS measures the similarity between any two classes $m_k, m_l \in \mathbf{M}$ of the taxonomy (as indicated for the tokens of taxonomy class m_k and m_l in Fig. 4). Opposed to WCD, for tokens (words) in BCS, we would like to have the dissimilarity between vectors of tokens from different taxonomy class to be as high as possible. For example, we consider the similarity measure between the class 'mechanical failure' (m_k) and 'material failure' (m_l) is given by averaging the cosine similarity between ('leak'$\in m_k$, 'corrosion'$\in m_l$); ('leak'$\in m_k$, 'erosion'$\in m_l$); ('vibration'$\in m_k$, 'corrosion'$\in m_l$) $\cdots$. Equation (5) describes the mathematical expression for measuring the between class similarity. Here, $\mathbf{v_{w_{T,m_{k_r}}}}$ is the embedding vector for r^{th} word in taxonomy class m_k and $\mathbf{v_{w_{T,m_{l_s}}}}$ is the embedding vector for s^{th} word in taxonomy class m_l.

$$BCS = \sum_{\substack{\forall m_k in M \\ m_l \in M}} \sum_{\substack{w_{T,m_{k_r}} \in m_k \\ w_{T,m_{l_s}} \in m_l}} (\frac{\mathbf{v_{w_{T,m_{k_r}}}} \cdot \mathbf{v_{w_{T,m_{l_s}}}}}{||v_{w_{T,m_{k_r}}}|| \, ||v_{w_{T,m_{l_s}}}||}) \quad (5)$$

As we propose to learn the semantic and taxonomic information simultaneously, the tokens (words) of the taxonomic class borrow their embedding vectors from the same weight vector of the original SG model. The similarity measures are now combined with the original NCE loss for simultaneous learning of efficient word embeddings. The new loss function is called as the Custom Word Embedding Model loss (CWEM loss) and is given by equation (6) and is defined as a weighted average (weights given by α) of the NCE Loss and the similarity measures given by (WCD and BCS).

$$CWEM Loss = \alpha \times NCE\ Loss + (1-\alpha) \times \{WCD + BCS\} \quad (6)$$

The simultaneous learning in CWEM takes information from SG architecture (shown Fig. 3) and from the WCD and BCS sets (shown in Fig. 4). The weighted average of both the losses gives a trade-off between the semantic and taxonomic information to be incorporated in the learned embedding. It allows the user to weigh the extent of taxonomic information the user finds essential to incorporate in the generated word representation. The lower the value of α, the higher the taxonomic information in the generated word representation.

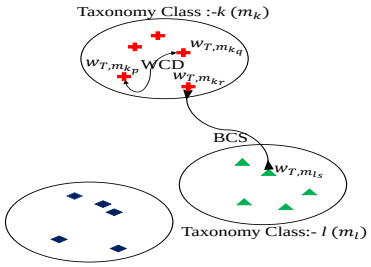


Fig. 4: Tokens in WCD and BCS sets of the CWEM.

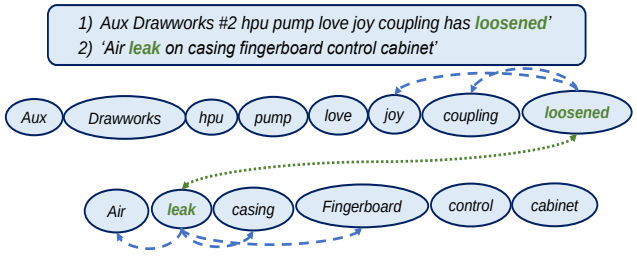


Fig. 5: Intuition for using two information sources for better representation. The dashed lines (– – –) represent learning through NCE loss for the semantic information while the dotted lines ($\cdots$) represent learning through the similarity of taxonomic tokens to incorporate taxonomic information.

As an illustration, consider the example in Fig. 5. For the first sentence, the word 'loosened' learns semantic information about its neighboring words due to the NCE loss. Similarly, for the second sentence, the word 'leak' learns semantic knowledge about its neighboring words. However, as there are no matching words in the sentences, the sentences would be dissimilar. To overcome this shortcoming, the external knowledge from the taxonomy is used in the CWEM to increase the similarity between the words 'loosened' and 'leak', thus making the two sentences similar to each other. The implementation of the CWEM is given in Algorithm 1.

IV. Case Study (Dataset From Oil Rigs)

A. Available Maintenance Records

For our analysis, we use data obtained from eight systems (like High Pressure Mud System (mud-pump), Drill Rig Hosting System etc.,) available across 31 oil rigs. The maintenance

Algorithm 1 Initializing and implementing CWEM for a batch

Input = $\alpha, N, window\_size, num\_skips, lr, \mathbf{w_I} \in V$

▷ *lr is the learning rate for the Gradient Descent*

1: Initialize weights for different layers
   First: $\mathbf{v_{w_I}} \in W_{|V| \times N} \rightarrow \mathcal{N}(0,1)$
   Hidden: $\mathbf{h} \rightarrow \mathbf{v_{w_I}}^T$
   Output: $\mathbf{v}^{'}_{\mathbf{w_O}} \in W^{'}_{N \times |V|} \rightarrow \mathcal{N}(0,1)$
2: Create the WCD sets, having words from same taxonomy class $m^{'} \in \mathbf{M}$ in same set
   $WCD \rightarrow \{w_{T,m_{k_p}}, w_{T,m_{k_q}} \in m^{'} \; \forall (m_k \in \mathbf{M})\}$
3: Create the BCS sets, having words from different taxonomy classes $m_k, m_l \in \mathbf{M}$.
   $BCS \rightarrow \{w_{T,m_{k_r}} \in m_k, w_{T,m_{l_s}} \in m_l \; \forall (m_k, m_l \in M)\}$
4: Calculate $NCELoss$ using equation 3
5: Calculate WCD using equation 4
6: Calculate BCS using equation 5
7: $CWEMLoss = \alpha \times NCELoss + (1-\alpha) \times \{WCD + BCS\}$
8: $lr\_ex\_decay \rightarrow lr \times decay\_rate^{\{\frac{global\_step}{decay\_step}\}}$
9: Optimizer $\rightarrow$ $Gradient\_Descent$ (minimize $CWEMLoss$ using lr_ex_decay)

records shown in Fig. 1 are a representative sample of the dataset. We have 11682 maintenance logs (L), and the size of the vocabulary (V) is 12987.

B. Experimental Settings

For the case study, we demonstrate the application of our proposed model using two different Settings. In Setting 1, we use the failure mechanism taxonomy and try to cluster maintenance records that are associated with similar failure mechanisms. For Setting 2, we use the hierarchical equipment taxonomy where the equipment is branched into various sub-units which are further branched into maintainable items and parts. In Setting 2, we try to cluster maintenance records that are associated with the same sub-units, maintainable items or parts.

1) Setting 1: On Basis of Failure Mechanism

Here, our aim is to identify clusters of maintenance records which describe maintenance events caused by similar failure mechanisms. The failure mechanism taxonomy is adapted from [42] and a few sample rows are shown in Table I . We have five failure mechanisms in the taxonomy as denoted in [42]. The number of terms in the taxonomy are summarized in Table II. We consider uni-gram (single word) tokens for this Setting. Tokens in the taxonomy are processed using steps demonstrated in Fig. 6a. The lementized tokens are obtained using a third party package in Python. For any tokens that are not accurately converted, we manually add extra tokens which represent the base form of the words (e.g., for the token 'Leakage', we also add 'Leak'). The taxonomy tokens are then combined with each other to form elements of the WCD set which are pairwise combination of tokens from the same class. Similarly, the BCS set is formed by pairwise combination of tokens from the same class. The formation of the sets is illustrated in Fig. 6b.

2) Setting 2: On Basis of Mud-Pump Equipment Taxonomy

In this Setting, our aim is to identify the cluster of maintenance records which describe maintenance activities concerned with same sub-units (or constituents components) present in the mud-pump taxonomy. The sample of the mud-pump taxonomy is shown in Table III. The taxonomy is created as a bill of materials (BOM) for the equipment. As the current model incorporates information from a single hierarchy taxonomy, we collapse the multi-hierarchy taxonomy of mud-pump to a single hierarchy by collapsing the maintainable items and parts of each sub-unit into a set which we term as sub-parts. We have five classes in the equipment taxonomy. The number of terms present in each class are shown in Table IV. The processing step 7a for the taxonomy has an additional step as there are many sub-parts which have multiple words in them and are thus converted appropriately into n-grams before training the model. Word tokens present in multiple sub-branches are removed so that each token can have membership exclusively to a single class. Sets of tokens required to develop the model are then created in the similar fashion as discussed in Section IV-B1 and shown in Fig. 7b.

TABLE I: A sample of taxonomy obtained from [42]

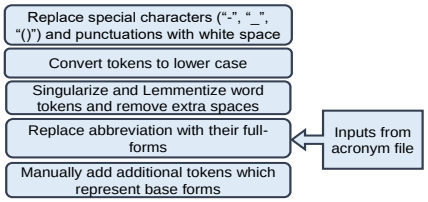
Failure Mechanism	Subdivision	Description
Mechanical	Leakage	External and internal leakages, either liquids or gases. If the failure mode at equipment unit level is leakage, a more causal-oriented failure descriptor should be used wherever possible
	Vibration	Abnormal vibration. If the failure mode at equipment level is vibration, a more causal-oriented failure descriptor should be used wherever possible
..	..	..

TABLE II: Number of terms in failure mechanism taxonomy groups

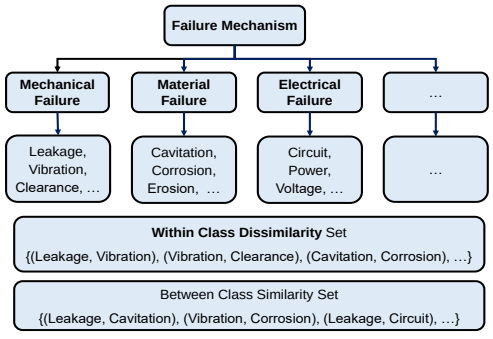
Failure mech.	Mechanical	Material	Hydraulic	Electrical	Control
# of tokens	28	25	10	10	6

C. Experimental Setup

We train the word embeddings using the CWEM for each Setting separately and generate separate word embeddings for each Setting. As shown in Algorithm 1, the embedding vector for each word is initialized using a standard normal distribution. To train the algorithm, we choose the word embedding lengths $N = 100$. The context window length is



(a) Processing failure mechanism taxonomy.



(b) Formation of sets for Setting 1.

Fig. 6: Setting 1 data preparation.

TABLE III: A sample of equipment taxonomy for mud-pump

Equipment Unit	Sub-Unit	Maintainable Item	Parts
Mud-Pump	Fluid End	Manifold	Discharge Manifold
			Suction Manifold
		Piston and Liner	Piston

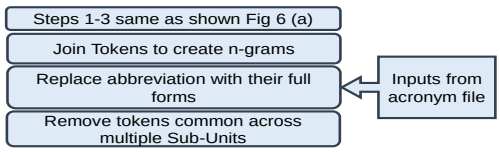
TABLE IV: Number of terms in equipment taxonomy groups

Sub-units	Drive Motor	Fluid End	Motor Cooling	Power End	Tool Control
# of tokens	10	22	14	19	10

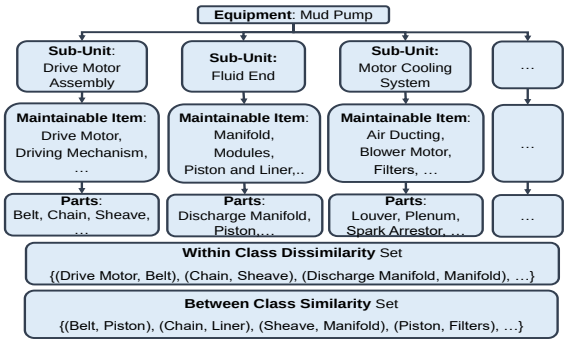
chosen to be 10 ($C = 10$; 5 context/target words to the left and right of the input word) and the batch size is chosen to be 256. We set the hyperparameter $num\_skips = 8$ to specify the number of words to be randomly sampled from the context window while learning embedding. To exponentially decay the learning rate of the gradient descent optimization, a decay step of 200 and a decay rate of 0.875 is selected. We use three competitive models to compare the performance of clustering of our method as described below:

1) Competing Models

1) **CWEM with α = 1/SG model**: The CWEM with α= 1.0 represents the original skip-gram model in our setting.
2) **GoogleNews**: Google's set of global word embeddings which contain pre-trained word embeddings developed on several Google news article stored in the GoogleNews-vectors-negative300.bin.gz
3) **Attract-Repel Model**: The Attract-Repel model proposed by [18] uses pre-trained word embeddings and incorporates additional information regarding the pairs of words which are either synonyms or antonyms. Please note that, for Attract-Repel model, elements of WCD set forms synonym set while elements of the BCS set forms antonyms set.
4) **Dict2Vec**: The Dict2Vec model presented in [33] learns embedding for various words using dictionary definitions provided in multiple dictionaries like Cambridge, Oxford, etc., to incorporate the meaning of each word while learning word embeddings. We utilize the pre-trained word embeddings for Dict2Vec with embedding size of 100 and also use Dict2Vec as a base embedding for Attract-Repel Model.
5) **Joint Rep using GloVe**: The joint learning model that we incorporate is proposed in [19], [20]. The authors proposes to use a single step joint learning model which incorporates the contextual information by using the loss function for GloVe [21]. The GloVe loss function is constructed using a term co-occurrence matrix which incorporates terms that co-occur with each other in a maintenance record. The taxonomic knowledge is incorporated by minimizing the distance between the terms that belong to the same taxonomy branch and also co-occur with each other in the record.



(a) Processing mud-pump taxonomy.



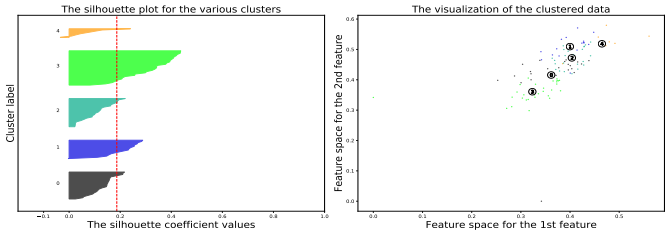
(b) Formation of Sets for Setting 2.

Fig. 7: Setting 2 data preparation.

2) Implementation

A clustering experiment is conducted by forming clusters of the selected documents (maintenance records) using the competitive models and CWEM with $\alpha \in \{0.2, 0.35, 0.5, 0.65, 0.8 \text{ and } 1.0\}$. To perform the experiment, documents from the processed corpus are selected for the two different Settings. We select a random sample of $L = 100$ documents for Setting 1. We manually identify the true failure mechanism indicated by each document and mark it as the true label for the given record. For Setting 2 as well, we select a set of $L = 100$ documents indicating the sub-unit which was repaired for the mud-pump. The sub-parts and sub-units for documents in Setting 2 can be assumed to occur primarily as 'nouns' in the documents, and hence we filter the selected documents to only include 'noun' and 'verb' before performing the experiment in order to remove noisy words. Such Part of Speech ('POS') targeted filtering is not feasible for Setting 1 as failure mechanisms could also occur as adjectives describing the condition of the part requiring maintenance.

Next, to cluster the documents in each dataset, the pairwise distance matrix is generated for maintenance records using word embeddings from each model (competitive and CWEM). The pairwise distance matrix measure is the pairwise distance between documents contained in the dataset and is a square matrix of dimension $(L \times L)$ for each dataset. To generate the pairwise distance matrix, one approach would be to just simply average the word embeddings of all words in the maintenance record and calculate the distance between these averaged embeddings. However, this will be very naive and would not be able to incorporate the essential information in the maintenance records because of the noise present in the dataset. To overcome this limitation, the pairwise distance between documents is measured using the *Words Mover Distance* (WMD) algorithm proposed by [43]. The WMD distance measures the dissimilarity between two text documents as the minimum amount of distance that the embedded words of one document need to "travel" to reach the embedded words of another document. The distance matrix is then supplied as an input to the *k*-means algorithm in order to determine the cluster label for each document. We use *k*-means because we have predetermined numbers of clusters for our experiment. Further, the *k*-means algorithm is highly efficient having comparable performance with other available clustering algorithms [39]. Using the predicted label from *k*-means and the manually marked label for each document



(a) Silhouettes analysis for CWEM with α = 0.65 model

(b) Silhouettes analysis for SG (CWEM with α = 1.0) model

(c) Silhouettes analysis for Attract-Repel (SG) Model

Fig. 8: Comparison of silhouettes analysis for CWEM with α = 0.65 model, SG (CWEM with α = 1.0) model and Attract-Repel(SG) model

evaluation metrics are generated.

D. Evaluation Criteria and Metrics

To evaluate the models' performance, we use the Adjusted Ranked Index (ARI) and the Silhouttes score (SSc) as the metric. The ARI is a similarity measure between two clustering schemes which considers all pairs of samples and the count of pairs that are assigned in the same or different clusters in the predicted and true clustering scheme. The ARI is calculated between the predicted label and manually marked label. High ARI would mean that high association exists between the predicted labels and the true labels. The SSc measures the consistency of the clusters formed and indicates how well the element is matched to its own cluster as opposed to the neighboring clusters. The range of the SSc is [-1, 1], and the higher the SSc, the better the efficiency of the clusters formed.

E. Results

The ARI and SSc scores obtained from the clustering analysis are discussed here. Fig. 8 is generated during silhouettes analysis for the CWEM with α = 0.65, SG model and Attract-Repel (SG) model. The plots on the left panels demonstrate the thickness of the cluster formed and the plots on the right panel indicate the clustered points on a 2-dimensional space. It can be seen from the plots on the right panel the clusters for CWEM with α = 0.65 model are well separated as compared to other competitive models. Also, the average SSc (indicated by dashed red line in the left panel) is higher for CWEM with α = 0.65 model. The training time for 100 batches of Attract-Repel model with Skip-Gram embeddings is 18 seconds, whereas, it took 0.2 seconds to train 100 batches of CWEM.

TABLE V: Comparison of model performance

	Setting 1		Setting 2	
Model Name	**ARI**	**SSc**	**ARI**	**SSc**
CWEM $\alpha = 0.20$ scaled	0.211	0.104	0.191	0.136
CWEM $\alpha = 0.35$ scaled	0.295	0.104	0.321	0.136
CWEM $\alpha = 0.50$ scaled	0.336	0.105	0.342	0.139
CWEM $\alpha = 0.65$ scaled	0.338	0.106	0.42	0.187
CWEM $\alpha = 0.80$ scaled	0.356	0.098	0.423	0.181
CWEM $\alpha = 1.0$ scaled/SG	0.028	0.079	0.096	0.093
Attract-Repel (SG)	0.059	0.081	0.17	0.105
Google News	0.052	0.09	0.116	0.091
Attract-Repel (Google News)	0.106	0.087	0.129	0.094
Dict2Vec	0.088	0.048	0.141	0.089
Attract-Repel(Dict2Vec)	0.155	0.064	0.131	0.088
Joint Rep using GloVe	0.073	0.084	0.132	0.122

The results for both the Settings of all the models are summarized in Table V. It can be observed that for a random sample of 100 documents the ARI and SSc for all CWEM with $\alpha < 1.0$ is better than the competitive (Attract-Repel, Dict2Vec, Glove) and baseline (word2vec, Google News) models. The results for the competitive (Attract-Repel, Attract-Repel (Dict2Vec), Joint Rep using GloVe) model are however better than their corresponding baseline models. For both the Settings, it can be observed that ARI and SSc is better for $\alpha > 0.5$. The ARI and SSc for Setting 2 are highly distinct for CWEM as compared to the competitive models. This distinction can be attributed to the POS filtering carried out on the documents for Setting 2 which results in a higher frequency of taxonomy terms in the documents as compared to the other terms. The CWEM with $\alpha = 0.65$ performs better in terms of SSc for both the Settings. The Joint Rep using GloVe model does not perform well in the experiments. This can be attributed to the requirement of co-occurring taxonomic terms in the maintenance records. The Attract-Repel (Dict2Vec) model performs pretty well for Setting 1 as taxonomy terms in Setting 1 contains terms like 'leak', 'corrosion' etc., which are generic in nature and have the same contextual meaning in various English dictionaries. However the model performance decreases when a more specific industrial taxonomy is used as can be seen in Setting 2 results. To better understand the convergence of ARI for each model, as well as the consistency of the models, we consider bootstrapping next.

For bootstrapping, we use the same dataset of selected 100 documents from which we create a balanced set of 50 documents where we sample $L = 10$ documents of each class from the initial set. This process is repeated for $B = 100$ iterations, and in each iteration, we calculate the ARI for the sampled set. The mean value of the ARI is calculated to analyze the performance of each model and is tabulated in table VI. By observing the bootstrapped results, we infer that, for documents having high frequency of non-taxonomy tokens (as in the case for maintenance records of Setting 1 and Setting 2), it is better to incorporate semantic knowledge to a higher extent by keeping $0.5 < \alpha < 1.0$ to allow for better clustering. However, when it is intuitive that the number of taxonomy tokens would occur at high frequency in the documents under consideration (as in Setting 2), the model performs well even when taxonomic information is incorporated at a higher extent with values of $0.2 < \alpha < 0.5$. We illustrate the results obtained

for bootstrapping in Fig. 9. It can be seen that the mean of bootstrapped ARI for CWEM with $\alpha = 0.65$ is significantly larger than the mean of the other competitive models.

TABLE VI: Results from bootstrapping

Model Name	Mean ARI Setting 1	Mean ARI Setting 2
CWEM $\alpha = 0.2$	0.1263	0.2669
CWEM $\alpha = 0.35$	0.1282	0.3689
CWEM $\alpha = 0.5$	0.1268	0.3299
CWEM $\alpha = 0.65$	0.1513	0.3605
CWEM $\alpha = 0.8$	0.1398	0.3560
CWEM $\alpha = 1$ scaled/SG	0.0436	0.1742
Attract-Repel (SG)	0.1000	0.2506
Google News	0.0863	0.1188
Attract-Repel (Google News)	0.1251	0.1330
Dict2Vec	0.0882	0.1405
Attract-Repel (Dict2Vec)	0.1452	0.1509
Joint Rep using GloVe	0.0986	0.1308

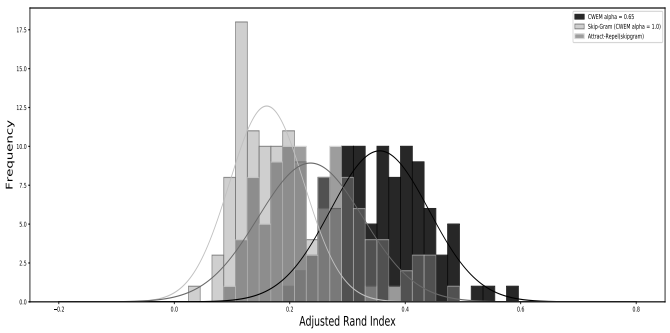


Fig. 9: Bootstrapping results for CWEM with $\alpha = 0.65$, SG (CWEM with $\alpha = 1.0$) and for Attract-Repel (SG) model

V. Conclusion and Future Directions

In this paper, we have proposed a novel distributed representation of textual description available in maintenance records. We use information from two sources (namely semantic information and taxonomic information) to efficiently learn the word distribution, and a weighting parameter governs the learned representation. In terms of identifying clusters, our proposed methodology using CWEM outperforms other models. The model also demonstrates that simultaneous incorporation of taxonomic and contextual/semantic information helps in developing efficient numerical representations for words as compared to the traditional two-step procedure. There are two key observations in this regard: First, the classification of documents depend on the secondary source of information, i.e., the clusters formed will be influenced by the taxonomy which has been used. Second, the parameter α provides a simple way to control the degree of influence exerted by the secondary source of information.

In practice, for a new dataset, the parameter $\alpha = 1$ would yield clusters by using information present only within the documents, thus providing insights about the groups from a general perspective. By gradually decreasing the value of α in the presence of an available taxonomy, the model will identify clusters based on the taxonomy. Therefore, the clusters will be similar based on the provided information. In this manner, the way of combining different sources of information to identify groups would yield similarities or differences with respect to the provided taxonomy. That is, two documents can belong to one cluster with respect to one taxonomy, while they may belong to different clusters with respect to another taxonomy.

Our work can be used by OEMs in a variety of applications. For example, the clusters of documents in Setting 1 would represent records from different failure mechanisms, this information can help OEMs to better stratify their event data before performing reliability studies. For example, the authors in [8], analyze clinical notes of patients for unit-matching in survival analysis. In industrial domain a similar approach can be used for identifying stratification in reliability data using the clusters obtained by CWEM in Setting 1. For Setting 2 the clusters represent different components or sub-units of the equipment that required maintenance, thus, clustering them can help OEMs have targeted improvements in equipment design or changes to their warranty policies or spares parts inventory management. For future research, the current model could be extended to incorporate information about the lexical parent-child relations present in multi-hierarchy taxonomies in a straightforward manner without having the dependency on additional resources/supervision.

References

[1] J. Wan, S. Tang, D. Li, S. Wang, C. Liu, H. Abbas, and A. V. Vasilakos, "A manufacturing big data solution for active preventive maintenance," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2039–2047, 2017.

[2] D. Rajpathak and S. De, "A data-and ontology-driven text mining-based construction of reliability model to analyze and predict component failures," *Knowledge and Information Systems*, vol. 46, no. 1, pp. 87–113, 2016.

[3] F. Wang, T. Xu, T. Tang, M. Zhou, and H. Wang, "Bilevel feature extraction-based text mining for fault diagnosis of railway systems," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 1, pp. 49–58, 2016.

[4] J. P. Usuga Cadavid, B. Grabot, S. Lamouri, R. Pellerin, and A. Fortin, "Valuing free-form text data from maintenance logs through transfer learning with camembert," *Enterprise Information Systems*, pp. 1–29, 2020.

[5] M. Alkahtani, A. Choudhary, A. De, and J. A. Harding, "A decision support system based on ontology and data mining to improve design using warranty data," *Computers & Industrial Engineering*, vol. 128, pp. 1027–1039, 2019.

[6] K. Rajbabu, H. Srinivas, and S. Sudha, "Industrial information extraction through multi-phase classification using ontology for unstructured documents," *Computers in Industry*, vol. 100, pp. 137–147, 2018.

[7] M. P. Brundage, T. Sexton, M. Hodkiewicz, A. Dima, and S. Lukens, "Technical language processing: Unlocking maintenance knowledge," *Manufacturing Letters*, vol. 27, pp. 42–46, 2021.

[8] R. Mozer, L. Miratrix, A. R. Kaufman, and L. J. Anastasopoulos, "Matching with text data: An experimental evaluation of methods for matching documents and of measuring match quality," *Political Analysis*, vol. 28, no. 4, pp. 445–468, 2020.

[9] M. Hodkiewicz and M. T.-W. Ho, "Cleaning historical maintenance work order data for reliability analysis," *Journal of Quality in Maintenance Engineering*, 2016.

[10] K. Saetia, S. Lukens, X. Hu, and H. Pijcke, "Data-driven approach to equipment taxonomy classification," in *Proceedings of the PHM Society Conference*, 2019.

[11] E. Khabiri, W. M. Gifford, B. Vinzamuri, D. Patel, and P. Mazzoleni, "Industry specific word embedding and its application in log classification," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2713–2721.

[12] S. Sia, A. Dalmia, and S. J. Mielke, "Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too!" *arXiv preprint arXiv:2004.14914*, 2020.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st International Conference on Learning Representation (ICLR 2013)*, pp. 1–12, 2013.

[14] Z. Yu, H. Wang, X. Lin, and M. Wang, "Learning term embeddings for hypernymy identification," in *Proceedings of the 24th International*

Conference on Artificial Intelligence, ser. IJCAI'15. AAAI Press, 2015, p. 1390–1397.

[15] I. Vulić and N. Mrkšić, "Specialising word vectors for lexical entailment," *arXiv preprint arXiv:1710.06371*, 2017.

[16] C. Zhang, F. Tao, X. Chen, J. Shen, M. Jiang, B. Sadler, M. Vanni, and J. Han, "TaxoGen: Unsupervised Topic Taxonomy Construction by Adaptive Term Embedding and Clustering," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 2701–2709.

[17] A. T. Luu, Y. Tay, S. C. Hui, and S. K. Ng, "Learning term embeddings for taxonomic relation identification using dynamic weighting neural network," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 403–413.

[18] N. Mrkšić, I. Vulić, D. Ó. Séaghdha, I. Leviant, R. Reichart, M. Gašić, A. Korhonen, and S. Young, "Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints," *Transactions of the association for Computational Linguistics*, vol. 5, pp. 309–324, 2017.

[19] M. Alsuhaibani, D. Bollegala, T. Maehara, and K.-i. Kawarabayashi, "Jointly learning word embeddings using a corpus and a knowledge base," *PloS one*, vol. 13, no. 3, p. e0193094, 2018.

[20] D. Bollegala, M. Alsuhaibani, T. Maehara, and K.-i. Kawarabayashi, "Joint word representation learning using a corpus and a semantic lexicon," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[21] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[24] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[25] Q. Chen, X. Zhu, Z.-H. Ling, D. Inkpen, and S. Wei, "Neural natural language inference models enhanced with external knowledge," *arXiv preprint arXiv:1711.04289*, 2018.

[26] R. Li, Z. Jiang, L. Wang, X. Lu, M. Zhao, and D. Chen, "Enhancing transformer-based language models with commonsense representations for knowledge-driven machine comprehension," *Knowledge-Based Systems*, p. 106936, 2021.

[27] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "Ernie: Enhanced language representation with informative entities," *arXiv preprint arXiv:1905.07129*, 2019.

[28] M. E. Peters, M. Neumann, R. L. Logan IV, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, "Knowledge enhanced contextual word representations," *arXiv preprint arXiv:1909.04164*, 2019.

[29] R. Wang, D. Tang, N. Duan, Z. Wei, X. Huang, C. Cao, D. Jiang, M. Zhou *et al.*, "K-adapter: Infusing knowledge into pre-trained models with adapters," *arXiv preprint arXiv:2002.01808*, 2020.

[30] Y. Ma, J. Zhao, and B. Jin, "A hierarchical fine-tuning approach based on joint embedding of words and parent categories for hierarchical multi-label text classification," in *International Conference on Artificial Neural Networks*. Springer, 2020, pp. 746–757.

[31] A. Roy, Y. Park, and S. Pan, "Predicting malware attributes from cybersecurity texts," *UMBC Student Collection*, 2019.

[32] T. B. Sexton and M. P. Brundage, "Nestor: A tool for natural language annotation of short texts," *J. Res. NIST*, vol. 124, 2019.

[33] J. Tissier, C. Gravier, and A. Habrard, "Dict2vec: Learning word embeddings using lexical dictionaries," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 254–263.

[34] C. Malon, "Overcoming poor word embeddings with word definitions," *arXiv preprint arXiv:2103.03842*, 2021.

[35] M. Alam, R. Biswas, Y. Chen, D. Dessì, G. A. Gesese, F. Hoppe, and H. Sack, "Hierclassart: Knowledge-aware hierarchical classification of scholarly articles," in *Companion Proceedings of the Web Conference*, 2021.

[36] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Neural Information Processing Systems (NIPS)*, 2013, pp. 1–9.

[37] T. K. Landauer and S. T. Dumais, "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge," *Psychological Review*, vol. 104, no. 2, pp. 211–240, 1997.

[38] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning research*, vol. 3, no. 4-5, pp. 993–1022, 2003.

[39] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues, "Clustering algorithms: A comparative approach," *PloS one*, vol. 14, no. 1, p. e0210236, 2019.

[40] X. Rong, "word2vec parameter learning explained," *ArXiv*, vol. abs/1411.2738, 2014.

[41] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

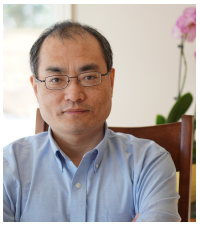
[42] B. ISO, "14224,"petroleum, petrochemicals and natural gas industries: collection and exchange of reliability and maintenance data for equipment"," *British Standards Institution, UK*, 2016.

[43] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *International conference on machine learning*, 2015, pp. 957–966.

Abhijeet Sandeep Bhardwaj received his Integrated M.Tech. degree in Geophysical Technology from the Indian Institute of Technology Roorkee, Roorkee, India, in 2017. He is working towards the Ph.D. degree at the Department of Industrial and Systems Engineering, University of Wisconsin–Madison, Madison, WI, USA. His research interests include unstructured natural language data mining and analytics, industrial prognosis using data fusion and decision making for Internet-of-Things-enabled smart and connected systems.

Akash Deep received the B.Tech. degree in production and industrial engineering from the Indian Institute of Technology Roorkee, Roorkee, India, in 2017 and MS in Statistics from University of Wisconsin–Madison, Madison, WI, USA in 2021. He is working toward the Ph.D. degree at the Department of Industrial and Systems Engineering, University of Wisconsin–Madison, Madison, WI, USA. His research interests include predictive analytics, survival modeling, and service decision making in Internet-of-Things-enabled smart and connected systems.

Dharmaraj Veeramani received the B.S. degree in mechanical engineering from the Indian Institute of Technology Madras, Chennai, India, in 1985, and the M.S. and Ph.D. degrees in industrial engineering from Purdue University, West Lafayette, IN, USA, in 1987 and 1991, respectively. He is the E-Business Chair Professor with the Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI, USA. His research focuses on emerging frontiers of digital business, Internet of Things technologies and applications, smart and connected systems, and supply chain management.



Shiyu Zhou received the B.S. and M.S. degrees in mechanical engineering from the University of Science and Technology of China, Hefei, China, in 1993 and 1996, respectively, and the master's degree in industrial engineering and the Ph.D. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, both in 2000. He is the Vilas Distinguished Achievement Professor with the Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI, USA. His research interests include data-driven modeling, monitoring, diagnosis, and prognosis for engineering systems with particular emphasis on manufacturing and after-sales service systems.