Combining natural language processing and multidimensional classifiers to predict and correct CMMS metadata

Arne Deloose[a,*], Glenn Gysels[b], Bernard De Baets[a], Jan Verwaeren[a]

[a] *KERMIT, Department of Data Analysis and Mathematical Modelling Ghent University, Coupure links 653, 9000 Ghent, Belgium*

[b] *Johnson & Johnson, Janssen Pharmaceutica NV, Turnhoutseweg 30, 2340 Beerse, Belgium*

Abstract

Computerized maintenance management systems (CMMSs) contain valuable data on the maintenance operations in an organization. A large part of these data consists of unstructured, written texts contained in failure notifications which are generated each time an unexpected failure occurs, enriched with structured metadata consisting of a number of labels that allow to categorize the failures, such as the type of failure, its cause or the corrective action that was taken. In this paper, we show that natural language processing techniques can be used to predict the structured metadata based on the unstructured text and even identify mislabeled notifications or ambiguous labels. Specific attention is given to the complexity that arises from the highly technical nature of the texts combined with a telegraphic writing style and heavy use of sentence fragments and abbreviations. Moreover, it is shown that exploiting dependencies between different components of the metadata, and regarding the prediction problem as a multidimensional classification problem, can improve the reliability of the predicted labels. We illustrate and test our label prediction pipeline on the CMMS data of a large pharmaceutical company.

Keywords: Natural Language Processing, CMMS, maintenance, failure logs, machine learning

*Corresponding author

Email addresses: arne.deloose@ugent.be (Arne Deloose), ggysels@its.jnj.com (Glenn Gysels), bernard.debaets@ugent.be (Bernard De Baets), jan.verwaeren@ugent.be (Jan Verwaeren)

1. Introduction

Of the many uses of industrial data, one of the most important ones is increasing the reliability and safety of systems. According to Cato and Mobley [6], maintenance costs can account for as much as 60% of controllable plant operating costs. These costs can be substantially reduced by improving maintenance policies based on data [20]. Historical data on failure events and maintenance logs, typically available in Computerized Maintenance Management Systems (CMMSs), are a potential source of information that can be exploited to optimize maintenance policies [4].

Failure notifications are arguably the most relevant type of data contained in a CMMS for optimizing a maintenance policy. Such notifications, of which an example is shown in Figure 1, contain a textual description of a failure event and are characterized by their technical nature, combined with a telegraphic writing style making heavy use of sentence fragments and abbreviations [11]. Because of their unstructured nature, they are often supplemented with structured metadata such as a date, time, location and a set of tags. Potential tags consist of an identifier indicating the asset to which the failure relates or one or more labels that are selected from predefined sets (such as failure codes). The structure that is present in the metadata and the tagging system allows the data to be mined efficiently and their summary statistics can, for instance, be used to gain insight into the frequency of particular types of failures and in some cases used to establish a preventive maintenance policy [2, 7, 13]. However, in practice, these metadata only provide a coarse description of the failure event and more fine-grained information is available in the less-structured textual description of the failure event. Therefore, an in-depth analysis of the unstructured text using natural language processing can reveal patterns that complement the information contained in the structured metadata (for instance topic modeling approaches such as latent Dirichlet allocation Roberto et al. [27]); or it can be used to automatically correct the metadata and identify ambiguous components. In this paper we address the latter and use natural language processing and machine learning techniques to predict part of the metadata based on the unstructured text. Specifically, we will focus on failure codes. A failure code is a label pertaining to a failure mode or type. By analysing which problem types occur most often, resources can be allocated optimally and policies can be adjusted. However, since these codes are the result of manual input (for instance selected from drop-down boxes when entering a notification in the CMMS, as illustrated in Figure 1) they are error-prone. Moreover, the set of available tags may vary over time and, as a result, a retagging of older notifications might be required when the CMMS is revised. In both cases, the (unstructured) free text that is available can be used to either flag potential errors in the input, or partly automate the retagging of old notifications. Given the number of failure notifications generated by a large production facility over time, manual correction and retagging of notifications is often not feasible. However, provided that

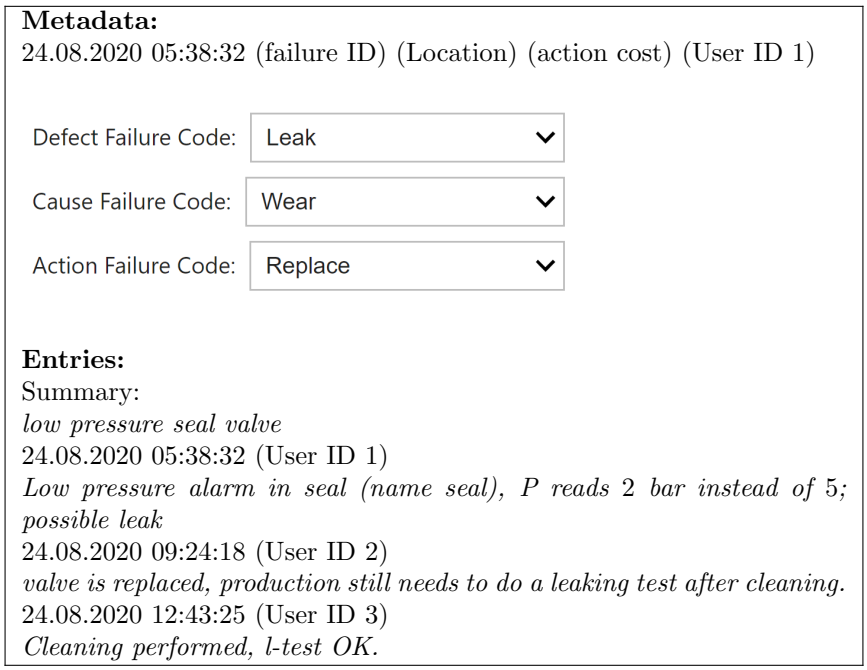

Figure 1: Example of a long form failure notification. Free text is italicized. This figure has been formatted for clarity.

sufficient correctly labeled notifications are available, a machine-learning-based classifier can be trained to automatically label or flag the remaining notifications using the free text data as input.

In the machine learning literature, the problem we are aiming to solve is called a multidimensional classification problem [24]. In contrast with traditional classification problems, where each object is only classified once and the output variable is a single nominal variable, a multidimensional classification problem consists of multiple classification problems, meaning that each object needs to be classified a (fixed) number of times, each time along a different dimension. In this problem, there are multiple nominal output variables for which a value needs to be predicted. For example, the failure notification in Figure 1 has been linked with three failure codes (defect, cause and action) and therefore all three codes should be predicted simultaneously. As we illustrate in this paper, using dedicated multidimensional classification methods can help to reduce the number of erroneous classifications. Another complicating factor is the type of language that is commonly used in the free-text fields of the notifications. The lack of a clear structure combined with a technical and telegraphic writing style complicates the automated analysis of the free text. This makes the problem different from more popular text categorization applications such as the categorization of social media posts [42] or newspaper articles [18].

A failure code and a free-text description of the failure can be seen as two sides of the same coin. Where the former is a structured way of describing a failure using predefined labels, the latter allows for more freedom in describing the same problem. Therefore, it is reasonable to assume that free text can be used to predict the failure codes. Nevertheless, in a broader maintenance context, other data sources could be used as well. A notable example are logs of supervisory control and data acquisition (SCADA) systems. It is reasonable to assume that a significant number of failures can be picked up by the sensors connected to a SCADA system. Retrospectively, these logs can be used to predict and/or correct failure codes and thus complement a free-text based approach. However, in our experience, unless dedicated sensors are installed to detect a specific failure, it is often challenging to link SCADA data to the root cause of a failure. The remainder of this paper is organised as follows. Section 2 discusses related work, while Section 3 introduces the case study that will be used to illustrate the methodology. Section 4 discusses the supervised methods for classification. In Section 5, the performance of the resulting tools is analysed using accuracy, F1-score, precision and recall, four common performance metrics used in supervised classification. Section 6 uses the best models to analyse individual labels, class imbalances and mislabeled notifications. Finally, Section 7 discusses the main conclusions and future perspectives.

2. Related work

Industrial production processes are monitored with increasing intensity. Even though most of these process data are captured for operational reasons such as process control, there is an increasing trend to use machine learning methods to learn from these historical process data and optimize plant operations. Most research on the use of machine learning models for maintenance focuses on the use of (numerical) sensor data. These models often rely on probabilistic fault prediction or the modelling of the degradation of a component [40, 43, 44]. Another option is to use an online real-time anomaly detection model to identify abnormal trends or patterns that could be caused by deteriorating components, as is used in Park et al. [21] on injection moldings. Alternatively, new data can be collected for the sole purpose of identifying faults, such as vibration data [29], thermographic imaging data [9], pressure or acceleration data [36], acoustic data [37] and ferrography data [28].

The use of machine learning methods to analyse CMMS data, and in particular the free-text data contained therein, is less studied in the context of maintenance. However, the field of natural language processing (NLP) has advanced to a point where automated analysis of free text is feasible. The rise of deep learning technologies has greatly improved the accuracy of NLP. As a result, classical applications of NLP such as sentiment analysis and opinion classification have become commonplace. NLP techniques can extract valuable informa-

tion from movie reviews [5], restaurant reviews [39] or customer opinions shared on social media [17]. Nevertheless, applications of NLP for processing CMMS data are less abundant. According to Devaney et al. [11], the application of NLP to industrial failure notifications is hampered by several specific properties of these data: (1) the use of inconsistent vocabulary; (2) the use of jargon and abbreviations; (3) a lack of public datasets; (4) a lack of well-formed input due to the presence of sentence fragments, spelling and grammatical errors; and (5) a lack of a direct link to the components of interest as it frequently occurs that the failure of a component is described without explicitly naming the component. These five properties apply to most industrial datasets. However, other properties can also apply depending on the branch of industry considered. In particular in the pharmaceutical production environment considered in our paper, additional aspects are: (a) the presence of temporal effects due to production shifts and the emergence of new drug products; (b) a large variety of assets, as a larger pharmaceutical production facility produces a large variety of drug products each requiring their own unit processes and equipment that can only be shared partly over products; (c) the presence of fleet aspects as some assets are present in multiplicate and can be interpreted as a more homogeneous fleet, but often with different usage patterns and age. Also, due to strict regulations regarding safety, even identical assets cannot be easily substituted in a production process without extensive validation. As assets cannot easily be interchanged, each asset will have a different usage history and will have vastly different loads applied to it during its lifetime, which could influence potential failures.

Despite the difficulties resulting from the aforementioned properties that distinguish failure notification texts from text used in more common NLP applications, several examples can be found in literature that use NLP on CMMS data. Zhang et al. [41] use free-text fields in failure notifications to predict associated failure codes and assess the correctness of the labels. The free-text fields are transformed into wordcounts (after preprocessing), which are then weighed according to their global frequencies (tf-idf weighing). Then, the SMOTE algorithm (Synthetic Minority Oversampling Technique) is used to balance the classes, followed by one of eight classification models, including support vector machines (SVMs), naive Bayes (NB) and Random Forests (RF). Arif-Uz-Zaman et al. [3] try to enrich the metadata in a CMMS by labeling downtime notifications as either resulting from a *failure* or *no failure*. This paper uses a less exhaustive pipeline, with a basic preprocessing and wordcount transformation (bag of words), followed by one of two models: SVM and NB. SVM outperformed the other models in both papers. In Tanguy et al. [31], SVM was also used to classify failure notifications. However, in this paper, both words and word fragments (substrings) were considered. This research showed that using all 3-character substrings combined with word stems showed a modest improvement over using only words, likely due to the presence of abbreviations, compound terms and alternate writing forms.

Apart from these traditional classification models, more advanced deep learning models can also be used to classify textual data. Such models are not commonly employed in the CMMS field, but they are popular in other NLP fields, such as the identification of hate speech [42] and sentiment analysis [8, 32]. Popular model choices for this type of problems are recurrent neural networks (RNN) using the GRU-implementation (Gated Recurrent Unit), Convolutional Neural Networks (CNN) and combined networks containing multiple layer types, such as a convolution layer followed by a GRU. The results of these models are usually quite similar and depend on the dataset at hand [38]. More recently, transformer models, a class of deep neural network models based on (self-)attention mechanisms, are surpassing the former state-of-the-art on many NLP applications [35]. The BERT (Bidirectional Encoder Representations from Transformers) model [12] is a prominent example of this class of models and has been used successfully to classify maintenance records (written in French) Usuga-Cadavid et al. [34]. Specifically, the task was to predict the duration and criticality of a maintenance issue to allow for better coupling of maintenance with production planning. A French variant of BERT (CamemBERT) was finetuned for this specific task. The results were compared with baseline methods and the finetuned CamemBERT model outperformed all the other models. This model was further developed to account for class imbalance [33].

3. A pharmaceutical failure notification dataset

Pharmaceutical production environments are highly controlled and regulated. The extensive documentation of all maintenance operations in a CMMS is common practice in such environments. Regulatory agencies also impose maintenance data collection for oversight purposes. In this particular case, the regulatory agencies require the archiving of 10-year backlogs on traceability (logging of replaced components) along with a wide range of other safety standards for critical machinery.

3.1. Description of the dataset

The data used in this paper has been extracted from the CMMS of a production site of Janssen Pharmaceutica in Geel, Belgium. The extract contains tens of thousands of failure notifications extracted over the last ten years (2010–2020), ranging from broken light bulbs to failures of large production assets such as industrial dryers. While these are called 'failure notifications', they are not all related to unexpected failures. Apart from the actual failures, there are also administrative notifications (usually for planned work such as installing a new sensor) and delay notifications. About one quarter of the notifications falls in one of the two latter categories. All notifications are written in Dutch. For

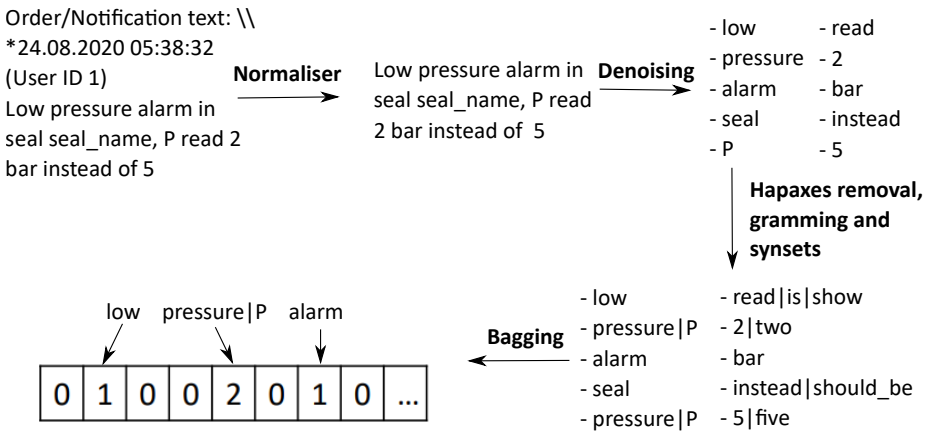

Figure 2: Example of a description being processed.

illustrative purposes, examples will be loosely translated into English in this manuscript.

Failure notifications are available as plain text files such as the one shown in Figure 1 and contain both structured metadata and unstructured text. The top lines contain the structured metadata. Each notification contains the following metadata: (1) a timestamp (date and time), (2) a unique ID, (3) a location code, (4) an action cost, (5) a user ID and (6) three failure codes. These failure codes are part of the input required by the CMMS. Each failure notification is labeled using three types of failure codes: a defect failure code depending on the type of defect of the failure (one of 16 codes is selected per notification); a cause failure code depending on the cause of the failure (one of 17 codes is selected per notification) and an action failure code depending on the action taken to address the reported problem (one of 15 codes is selected per notification). In Figure 1 these codes are suggestively represented by drop-down boxes. The (unstructured) free text consists of multiple fields: a first text field contains a short summary. The remaining text fields contain a more lengthy description, yet, have a telegraphic style with lots of sentence fragments and abbreviations. These text fields typically constitute multiple entries related to the same problem (with a total word count of around 150 on average). Basic problems usually have two or three entries (around 50 words) and are resolved in a matter of hours or days. More complex problems can take weeks and have up to six entries (around 300 words).

3.2. Preprocessing pipeline

The notifications cannot be processed in their raw form by most NLP methods. Therefore, a preprocessing pipeline was set up to transform the free text in the

failure notifications into a form that can be used as input for NLP methods. This pipeline is visually represented in Figure 2 and generally allows to transform a failure notification into a Bag-of-Words format. The *template removal* step removes all the text related to the fixed template that is used. A next step, loosely referred to as the denoising step, removes punctuation, performs case folding and uses the Dutch SpaCy language model [15] to remove stop words and perform lemmatization. Due to the technical nature of the text, a lot of words (one in three) are not known to the language model. Therefore, the SpaCy model was slightly extended with a set of additional more technical terms and abbreviations (user-specified). This step is followed by the *removal of hapaxes* and a *gramming* step (the linking of collocations, or N-grams). Traditional pretrained models such as the Google News Word2Vec model Mikolov et al. [19] are not suitable for the use case, because they lack the proper vocabulary and context. Instead, a new Word2Vec model was trained on the corpus using the Gensim implementation [26]. Due to the heavy usage of compound words in Dutch, a lot of multi-word English expressions (such as 'leaking test' and 'cleaning status') only require a single word in Dutch ('lektest' and 'reinigingsstatus'). Therefore, the maximum length of grams was set to two words. The next step is the *creation of synsets*, where synonyms are identified using OpenDutchWordNet [23] and treated as identical during further processing. The WordNet is also extended with more technical user-specified synonyms. The final step is the *embedding step*. In this step, the preprocessed notifications are mapped to fixed-length tensors. In the experiments described later, two embedding strategies are compared. The first embedding strategy is the Bag-of-Words (BoW) embedding where each *notification* is represented by a vector of word counts (with the length of the vector equal to the size of the vocabulary of the corpus), leading to a sparse representation. A variation on this basic embedding is the Term Frequency-Inverse Document Frequency (tf-idf) [16] embedding, which can easily be obtained from the BoW by reweighing. The weighing scheme uses the relative word count obtained by dividing the word frequency within a notification by the word frequency in the complete set of notifications. The former embeddings neither encode the order in which the words appear in the notification nor exploit information regarding the similarity of words. To resolve these potential limitations, a second embedding strategy uses a Word2Vec [26] algorithm (Gensim-implementation) to learn a word embedding that maps each word onto a k-dimensional vector (we set $k = 32$) such that words that often appear in the same context are mapped to similar vectors. To embed a notification, the embeddings of the words are computed and concatenated. To obtain fixed-length tensors, zero-padding is applied (we used a length of 387). In the experiments, we consider two variants to obtain word embeddings. A first variant uses Gensim Word2Vec to train the embedding from scratch using the full corpus of notifications. A second variant uses a pretrained Word2Vec model from Mikolov et al. [19] that is fine-tuned using Gensim on the corpus of failure notifications.

4. Methods

As part of the CMMS input, each failure notification is labeled using three types of failure codes: a defect failure code; a cause failure code and an action failure code. For a maintenance engineer, it is, for most notifications, rather easy to assign failure codes to a notification based on the free text. For reasons mentioned earlier, we investigate how this labeling process can be automated. In the machine learning literature this problem is known as a *multi-dimensional classification problem* [24], a problem where each data instance (in this case a notification) is associated with multiple (in our case three) multi-class labels. In the experiments that follow, we: (1) develop three base learners (one per label) that accept a document embedding as input and independently predict the three labels; and (2) investigate how the dependency between the labels can be exploited using methods from the literature on multi-dimensional classification.

4.1. Base learners

We compare several base learners. A first set of base learners relies on the BoW and tf-idf document embeddings obtained using the preprocessing pipeline discussed earlier (Section 3). These embeddings are used directly as input to a Random Forests classifier (RF), Support Vector Machines classifier (SVM) and a Naive Bayes classifier (NB). For the RF the number of variables considered at each split is chosen as the square root of the length of the embedding vector and the number of trees was set at 1000. The SVM uses a radial basis function kernel (with sigma equal to the inverse of the number of features multiplied by the variance) and the regularization constant was chosen using cross-validation. Models were fit using the Python package SciKit-Learn [22]. A second set of classifiers use a document representation that is obtained by concatenating the word embeddings of the individual words of the notifications. These representations are used as input to a Convolutional Neural Network (CNN), a Recurrent Neural Network that uses Gated-Recurrent Units (GRU) and a hybrid network that combines convolutional and recurrent layers (GRU+CNN). The architectures of these networks are inspired on the architectures that are found in Zhang et al. [42] for classifying Twitter messages (with lengths similar to those of failure notifications), and are presented in Figure 3. All models use the ADAM (ADAptive Moment estimation) optimizer for gradient descent with the categorical cross-entropy as loss function and a learning rate of 0.001, with 5 epochs for the CNN architectures and 10 for the other ones. The batch size used is 32. The GRU networks use a 10% dropout rate (these were the optimal hyperparameter settings found after parameter tuning). These networks were implemented and trained using Tensorflow [1]. Lastly, following the approach of [34], we used the RobBERT model [10] (a Dutch language model based on RoBERTa) as a neural network architecture. This transformer model was pretrained on OSCAR (Open Super-large Crawled Aggregated coRpus) [30], the largest Dutch corpus currently available (6.6 billion words). For our specific task, we finetuned

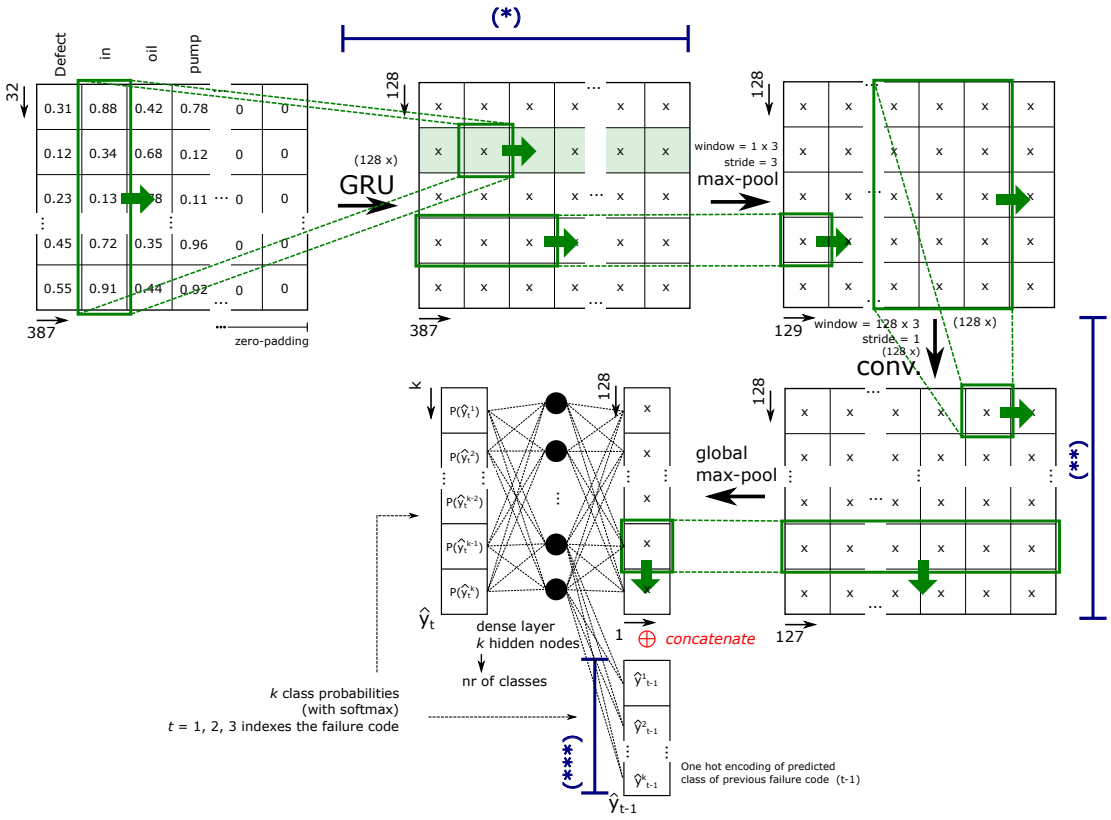

Figure 3: Architecture of the GRU$^{(*)}$+CNN$^{(**)}$ deep neural network used in the classifier chain. The architecture of the GRU network (without CNN layer) is obtained by replacing (**) with an additional (*) block. By replacing the (*) block with an additional (**) block, a pure GRU network is obtained. Finally, the base learners (without classifiers chains) can be obtained by removing (***).

the embeddings on the training set of CMMS texts using the same parameters as the other neural networks (ADAM optimizer, categorical cross-entropy loss function, learning rate 0.001, batch size 32 and 5 epochs). Both the neural networks and the BERT model use a final dense layer with softmax activation. This layer has the same number of neurons as there are labels in the output. In all cases, a train-test split was used (2/3 training, 1/3 testing). The training data were used to train the embeddings, fine-tune models and/or train the final classifiers. Performance was evaluated on the test set.

4.2. Exploiting dependencies

It is reasonable to assume that a dependency structure exists on the labels. For example, if for a given notification the defect code is *leak*, then it is highly un-

likely that the action code will be *cleaning*. By modifying the base classifiers, we can take these dependencies into account and produce classifiers that will, for instance, avoid predicting highly unlikely label combinations. Three strategies are compared to investigate the effect of incorporating label-dependencies. A first approach we consider, known as the *independent classifiers* approach [24], constructs an independent classifier for each label. As a result, these classifiers cannot learn a joint (conditional) distribution over the labels. As a second approach, we transform the three original labels into one super-label in which every unique label triplet becomes a new class. When considering all triplets of defect codes (16 classes), cause codes (17 classes) and action codes (15 classes), 4080 unique triplets are obtained. When considering each triplet as a separate class, a very large multi-class classification problem is obtained. This approach, known as the *class powerset* approach [24], does allow to learn a joint conditional distribution over the three labels. Each of the base learners discussed earlier was used to solve this multi-class classification problem. A third approach that we consider, relies on the use of *classifier chains* [24, 25]. Here, three base classifiers (one for each label) are trained sequentially (in our experiments the sequence is defect code, cause code and activity code). The first classifier is an independent base learner that predicts the defect code using the embedded text of the failure notification as input. The second classifier is a base learner that predicts the cause code but uses the predicted defect code (one-hot encoding) as an additional input (next to the embedded text). Lastly, the third classifier is a base learner that predicts the activity code and takes the predicted defect and cause codes as additional inputs. For the BoW and tf-idf embeddings, the predicted labels are added by concatenating them with the original embedding. For the sequential word embeddings, however, this is not possible, since every document is now a two-dimensional array rather than a one-dimensional vector. Therefore, the predicted labels are concatenated inside the neural network structures after the global max-pool (which reduces the tensors to one dimension), as illustrated in Figure 3.

4.3. Performance evaluation

To obtain an unbiased estimate of the performance of the models, the corpus is split in two-thirds as training set (training embeddings, fine-tuning language models and training multi-dimensional classifiers) and one-third as test set for all experiments. Moreover, to provide a multi-label view on the performance of the methods, we compute two types of classification accuracy. The first type is the *class accuracy* (CA), which is the average accuracy over the three labels. To define it formally, let $\mathbf{y}_i = (y_{i,1}, y_{i,2}, y_{i,3})$ be the observed triplet of labels for the ith failure notification. Moreover, let $\hat{\mathbf{y}}_i = (\hat{y}_{i,1}, \hat{y}_{i,2}, \hat{y}_{i,3})$ be the predicted triplet. Given a set of n notifications with d labels each (3 in our case), CA is defined as:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{d}\sum_{j=1}^{d}\mathbb{I}(y_{i,j}, \hat{y}_{i,j})\,,$$

where $\mathbb{I}(.,.)$ is the indicator function (takes value 1 if its arguments are equal and 0 otherwise). Note that CA evaluates the performance of each classifier independently and partial errors (instances for which not all labels are predicted correctly) still contribute positively to CA. The example accuracy (EA) is an alternative to the CA that does not account for partially correct predictions. For EA, a notification will only contribute to the accuracy if all three labels are (jointly) predicted correctly. EA is defined as:

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(\mathbf{y}_i, \hat{\mathbf{y}}_i)\,.$$

In addition to these measures of performance, also F1-score, precision (P) and recall (R) are reported. These are calculated as the weighed average of the measures for each class, using the one-versus-all method.

4.4. Short, intermediate and long texts

The free-text fields in a notification consist of a summary field followed by a sequence of entries which are ordered chronologically. Due to the structure of the text, some redundancy is often present (the message contained in the summary can often be derived from the individual entries). Moreover, the first few entries typically contain information on the type of problem that occurred and the remaining entries typically concern follow-up entries that contribute less to the identification of the defect, cause and action codes. If redundant or irrelevant information is present in the long texts, using shorter texts can be beneficial. To investigate this, we distinguish between (1) short texts that only contain the summaries; (2) intermediate texts that contain only the first user entry; and (3) long texts that contain all free text.

4.5. Experimental setup

The base learner, the technique used to exploit dependencies, the evaluation criterion, the size of the text that is used and the preprocessing pipeline can affect both the reported performance and the conclusions drawn from the analysis. To explore these combinations in a structured manner, we first explore the effect of combining different base learners and label dependence modeling strategies on the performance. Next, using the best-performing base learners, we investigate how the length of the text influences performance and perform an ablation analysis [14] of the preprocessing pipeline.

5. Results and discussion

5.1. Base learners and label dependency

The results of a first set of experiments are presented in Table 1. To obtain these results, the BoW embedding of the long texts was used for training the RF, NB and SVM models. The GRU, CNN and GRU+CNN models used the Word2Vec embedding of the long text, where the Word2Vec model was trained from scratch using the complete corpus and the dimension of the embedding was set at 32. When comparing the base learners, it can be concluded that, irrespective of the method that is chosen for exploiting label dependencies, the Random Forest classifier achieves the best performance among the BoW-based classifiers and outperforms the remaining classifiers by a large margin. Interestingly, when comparing the EAs of the Random Forest models, independent classifiers obtain an EA of 0.66, increasing to 0.75 when classifier chains are used. On the other hand, only a mild increase in performance is observed when comparing the CAs which are 0.88 and 0.90 respectively. Due to chaining, the accuracy of the individual base learners does not improve dramatically, but more often, entire label triplets are predicted correctly. This shows that the Random Forest classifier learns how to take conditional label dependencies into account. Additionally, these results show that deep learning architectures that rely on the Word2Vec embedding tend to outperform BoW-based base classifiers. With a CA of 0.95 and an EA of 0.93, the independent GRU-based architecture attains the best results. However, the networks that use convolutional architectures can almost match these accuracies. As the results are only mildly affected by the network structure, these results suggest that the Word2Vec embedding (at least to some extent) succeeds in learning useful relationships between words. Remarkably, chaining neural networks does slightly degrade performance for all architectures except for GRU+CNN. However, the effects are very small. Furthermore, we observe that using class powersets heavily degrades the performance of the base learners. Except for naive Bayes, all base learners perform extremely bad when using this approach. The evident explanation here is that the number of classes (4080 in this case) becomes too large given the number of observations. Most classes will only contain a handful of examples. Lastly, we observe that the RobBERT model obtains a slightly lower accuracy than the networks using convolutional and recurrent layers but outperforms the BoW-based models. It is generally assumed that the contextual embeddings that are derived by transformers are among the main reasons why they can outperform architectures using only convolutional and recurrent layers. We hypothesize that, as transformers typically need large datasets to learn these embeddings, there is insufficient data available to properly finetune the transformer network to allow it to outperform the other networks. Additionally, due to the highly specific technical language and telegraphic writing style, the texts that are used here differ rather strongly from the texts on which the RoBERT model was pretrained, reducing the efficiency of transferring the pretrained model to this

Base learner	Independent classifiers			Class powerset classifiers			Classifier chains		
	CA	**EA**	**F1**(EA)	**CA**	**EA**	**F1**(EA)	**CA**	**EA**	**F1**(EA)
Random Forest	0.88	0.66	0.63	0.82	0.63	0.56	**0.90**	**0.75**	**0.70**
Naive Bayes	0.43	0.08	0.06	0.51	0.20	0.16	0.47	0.13	0.12
Support Vector Machine	0.81	0.53	0.47	0.58	0.25	0.17	0.78	0.50	0.43
RNN (GRU)	**0.95**	**0.83**	**0.81**	0.38	0.08	0.01	0.91	0.79	0.79
CNN	0.94	0.79	0.78	0.34	0.08	0.02	0.86	0.69	0.67
GRU+CNN	0.94	0.79	0.76	0.51	0.19	0.09	0.93	0.83	0.82
RobBERT	0.90	0.73	0.74	0.42	0.12	0.04	0.87	0.77	0.78

Table 1: Base learners and multi-dimensional approaches.

new domain.

5.2. The effect of text length

Table 2 shows the results obtained using short, intermediate and long texts. Following the results obtained above, only chained Random Forest classifiers and independent GRU-based networks are used here. For the BoW-based Random Forest model, a distinction is made between the (unweighted) BoW embedding and the weighted variant tf-idf. For the GRU-based network, a distinction is made between a new embedding (Word2Vec model trained de novo) and a fine-tuned embedding (a Word2Vec model trained on Google News fine tuned on the corpus of notifications). The best performance is obtained for GRU-based networks that use the long text (CA = 0.95 and EA = 0.83), closely followed by the chained Random Forest classifier for texts of intermediate size (CA = 0.94, EA = 0.84). These results show that the short texts provide insufficient information to predict labels reliably (CA = 0.26 for Random Forests CA = 0.48 for GRU-based networks). Interestingly, extending the short summary with the first item of the long text field (to obtain the intermediate texts) allows to boost the performance significantly, and obtain among the best results (CA = 0.94 for both Random Forests and the GRU-based networks). Extending the texts further (long texts) leads to a degradation of the performance for the Random Forest classifier (CA drops from 0.94 to 0.90 and the EA drops from 0.84 to 0.75). The GRU-based network, on the other hand, attains a CA of 0.95 (the best CA in this study) and manages to improve the EA from 0.78 to 0.83. As mentioned earlier, the long texts extend the intermediate texts by considering additional items. Often, these items do not contain a lot of information that is relevant for the labeling of the notifications. The BoW representation used for the Random Forest does not retain the positional information, and lacks the potential to extract contextual information. Therefore, the model cannot learn to focus on the first part of the notification. In this application, this leads to a decreased performance of the Random Forest classifier. On the other hand, the GRU-based network can learn to focus based on position, and, as the results suggest, is not impacted negatively by the extension of the texts. On

the contrary, the additional information can be exploited to (marginally) boost the performance of the network.

Lastly, when comparing the embedding strategies, the BoW without weighing and the tf-idf embedding lead to identical performances. This suggests that only presence-absence information of the words is relevant when using this type of embedding. As the notifications are rather short and technical, word frequency is indeed less important as most words that signal the type of problem, its cause and the action that is taken, will often only appear once or, when they appear repeatedly, their frequency is not relevant for the classification problem. This observation is related to the telegraphic writing style which contrasts applications of NLP that focus on classifying more narrative texts, where using tf-idf often improves the performance. When comparing the Word2Vec embeddings, it seems that a *de novo* embedding strategy systematically outperforms the fine-tuning of an embedding that was pretrained on the Google News dataset [19]. We conclude that, for this application, fine-tuning the embedding does not succeed in transferring knowledge from related domains to the highly specific setting of notification analysis (CA = 0.95 for a *de novo* Word2Vec model versus CA = 0.91 for a fine-tuned model). An explanation for this observation can be found in the highly specific vocabulary that is used here. When considering the complete corpus of notifications, one in three words are not present in the pretrained Word2Vec model. Moreover, as the results suggest, the knowledge contained in the Word2Vec embedding of the words that do appear in the pretrained Word2Vec model does not contribute to the classification accuracy. This is in line with the underperformance of the RobBERT model that was observed before. As with the pretrained embeddings, the RobBERT model extensively relies on pretraining on external data and likely fails to transfer this (pretrained) knowledge to this rather different corpus.

5.3. Preprocessing: an ablation analysis

The bottom rows of Table 2 show the results of an ablation analysis of the preprocessing pipeline. Each row represents the accuracy that is obtained when omitting a single step from the preprocessing pipeline. In general, it appears that individual preprocessing steps only have a mild impact on the accuracy. The most substantial impact on the Random Forest classifier is caused by omitting digit removal, hapaxes removal and punctuation removal. Interestingly, gramming appears to have almost no effect on the performance of the models. It is important to note here that, in contrast with English, Dutch uses a lot of compounds. Therefore, there is less need to link collocations using gramming. It is possible that an English language corpus would benefit more from gramming. The limited impact of lemmatization can be linked to the fact that the descriptions are written in a telegraphic style which limits the number of conjugations that are used. This telegraphic style rarely uses stop words limiting the effect of stop word removal.

Text field	Preprocessing	Random Forest			RNN (GRU)		
		CA	**EA**	**F1 (EA)**	**CA**	**EA**	**F1 (EA)**
Short	BoW no weighing / *de novo* embed	0.26	0.03	0.03	0.48	0.04	0.01
	tf-idf / fine-tuned embed	0.26	0.03	0.03	0.44	0.03	0.01
Intermediate	BoW no weighing / *de novo* embed	**0.94**	**0.84**	**0.81**	0.94	0.78	0.77
	tf-idf / fine-tuned embed	0.94	0.84	0.81	0.91	0.76	0.75
Long	BoW no weighing / *de novo* embed	0.90	0.75	0.70	**0.95**	**0.83**	**0.81**
	tf-idf / fine-tuned embed	0.90	0.75	0.70	0.91	0.78	0.77
	Ablation analysis (No weighing):						
	- template removal	0.89	0.73	0.68	0.88	0.79	0.78
	- model stopword removal	0.90	0.76	0.71	0.94	0.78	0.77
	- manual stopword removal	0.90	0.75	0.70	0.95	0.82	0.80
	- whitespace removal	0.90	0.75	0.70	0.95	0.83	0.81
	- digit removal	0.87	0.68	0.63	0.93	0.74	0.72
	- punctuation removal	0.89	0.71	0.67	0.92	0.72	0.71
	- hapaxes removal	0.88	0.70	0.66	0.94	0.80	0.80
	- lemmatization	0.90	0.74	0.70	0.94	0.80	0.79
	- gramming	0.90	0.75	0.70	0.93	0.77	0.78

Table 2: Preprocessing ablation analysis and the effects of fine-tuned embeddings, BoW weighing schemes and choice of text field.

6. Practical considerations and applications of modeling results

In the previous section, a rigorous set of experiments was performed to find the best predictive models. In a follow-up analysis, we discuss a number of practical issues that arise when these models are used in practice for relabeling historical data or when model predictions are used to auto-complete the metadata fields. Most importantly, we show how analysing misclassifications can also help to optimize the maintenance policy.

6.1. Analysis of individual labels and class imbalances

Table 3 shows the performance results for each label separately. These results were obtained using the best Random Forest models (BoW-embedding on intermediate texts) and GRU-based models (de novo embedding on long texts). Additionally, this table shows the individual performances (Precision, Recall, F1-score and Accuracy) and the support (relative frequency) for the four most frequently occurring classes per label. The F1-scores and accuracies at the individual class level are close to one, showing that few misclassifications are present among the most popular classes. The combined support of these classes approaches or exceeds 80% for each label. As a result, the classification problem is highly imbalanced. Figure 4 presents a more complete view of the influence of class imbalances on the performance. From this figure, it is clear that classes

Code	Random Forest				Recurrent NNet (GRU)				Support	
	P	**R**	**F1**	**Acc**	**P**	**R**	**F1**	**Acc**	**%**	**Num**
Defect type (total)	0.94	0.95	0.94	0.95	0.95	0.95	0.95	0.95	100	1240
Mechanical failure	0.97	0.99	0.98	0.99	0.99	1.00	0.99	1.00	41	511
Measurement/control error	0.97	0.97	0.97	0.98	0.98	0.97	0.98	0.99	20	252
New work order (admi.)	0.90	1.00	0.95	0.98	0.99	0.98	0.98	0.99	15	184
Electrical	0.99	0.99	0.99	1.00	1.00	0.99	0.99	1.00	11	137
Cause (total)	0.91	0.89	0.89	0.89	0.90	0.91	0.91	0.91	100	1240
Defect component	0.80	0.97	0.88	0.93	0.98	0.97	0.98	0.99	25	310
Unknown	0.98	0.95	0.97	0.99	1.00	0.99	0.99	1.00	21	260
Operator error	0.99	0.88	0.93	0.98	0.99	0.98	0.98	1.00	17	209
Measurement/control error	0.94	0.90	0.92	0.92	0.97	0.99	0.98	1.00	14	176
Activity (total)	0.96	0.97	0.97	0.97	0.95	0.96	0.95	0.96	100	1240
Repair component	0.98	1.00	0.99	0.99	1.00	0.99	1.00	1.00	52	646
Replace component	0.99	1.00	1.00	1.00	0.99	1.00	0.99	1.00	22	270
Reset equipment	1.00	0.96	0.98	0.99	0.99	1.00	1.00	1.00	11	138
New work order (admi.)	0.94	0.92	0.91	0.93	0.94	0.97	0.95	0.96	4	52

Table 3: Precision (P), recall (R), F1-score (F1) and accuracy (Acc) for the four most abundant classes per failure code type for the best two models. Note that *New work order (admi.)* is a code that refers to an administrative action, rather than a traditional repair.

with a support that is in the range of 0–4% are predicted with variable accuracy. This observation holds for each of the three labels. The observation that less frequently occurring labels are susceptible to misclassification is not surprising but it clearly shows that one should be careful when relabeling historical data or using these models to auto-complete new notifications. In general, as previous results show, the relabeling will be quite accurate, but for some classes this will not be the case.

6.2. Analysis of mislabeled notifications

The notifications in the test set for which at least one label was predicted incorrectly by the GRU-based network (long texts, with *de novo* embedding) were examined in detail (17.0% of the test set). The free text was manually compared with the *true* labels assigned by the technicians and the predictions by the model. Based on this comparison, three groups of *errors* were identified: (1) *model errors* are disagreements between the predicted label and the label assigned by the expert where a closer inspection clearly indicates that the model predictions were wrong; (2) *label ambiguities* are disagreements where it is reasonable to believe that the label assigned by the technician is not the most appropriate label or where more than one class can describe that notification; and (3) *text ambiguities* are disagreements where also a human expert cannot re-identify the correct label based on the text; often these are notifications that lack a proper description of the problem at hand. This last class also includes

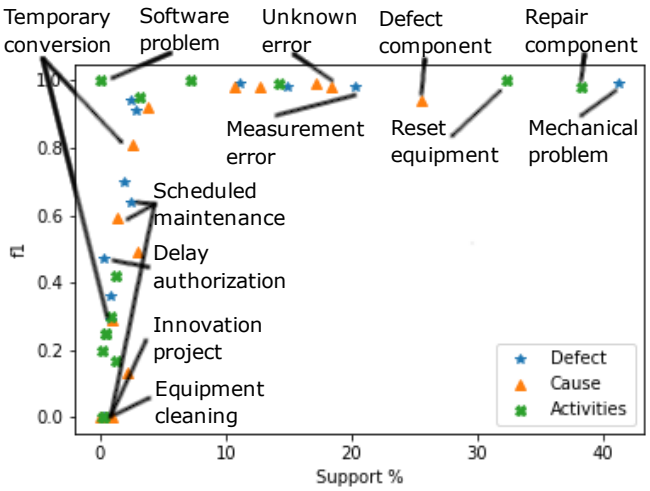

Figure 4: F1-score in function of the support for all classes. The best GRU model is used and the names of the top two, bottom two and median class are shown on the figure.

texts that reference information from external documents such as emails, or information that is conveyed through the labels themselves (there is no need to write 'replace' in the text if the label 'replace' was already given). These cases clearly illustrate that 'errors' are strongly dependant on the model used and its limitations. Nevertheless, they can serve as a useful starting point. Table 4 shows the distribution over these three groups. These results show that 43.6% of 17.0%, which is approximately 7.4%, of the notifications have at least one label that can be considered ambiguous. These notifications are particularly relevant to investigate more closely by a maintenance team as these are cases where a revision of the labels may be relevant. Moreover, in most cases that were assigned to the group of label ambiguity, multiple classes were considered relevant for a particular notification. Examining these ambiguous cases can help to optimize future labeling efforts by updating the set of available labels or (re)specifying the definition of a number of labels. Secondly, about 5.7% of the disagreements were cases where the textual description does not allow for a sensible labeling based on the written text. Here as well, the maintenance team can verify whether the policy regarding the use of these free text fields needs an update. Lastly, only 24.6% of the notifications for which at least one disagreement is observed can be attributed to model errors. This means that only 24.6% of 17.0%, which is approximately 4.2%, of the notifications was labeled incorrectly by the models.

Number of disagreements	Label ambiguities	Text ambiguities	Model errors	Total
3	0.008	0.016	0.008	0.032
2	0.055	0.024	0.032	0.095
1	0.373	0.294	0.206	0.873
Total	0.436	0.334	0.246	1.000

Table 4: Frequency distribution of the notifications with at least one disagreement between the model predictions and the labels assigned by the maintenance engineers, over three groups based upon manual re-inspection of the written text.

7. Conclusion

The research presented here, in addition to providing an operational solution to the needs of a specific pharmaceutical company, also yields several conclusions that can be generalized to data coming from other CMMSs.

Firstly, the performance of the multidimensional classification models was heavily dependent on a number of interconnected factors: base learner, label dependency and input length. The best models for our case were: (1) three independent GRU neural networks using a short *de novo* Word2Vec embedding using all available text (class accuracy: 0.95) and (2) a classifier chain Random Forest using a Bag-of-Words embedding using only the most relevant fraction of the text (class accuracy: 0.94). This indicates that for settings with shorter, highly relevant input text, Bag-of-Words models which exploit label dependencies are likely the optimal solution (assuming sufficient information is present in the input data). In contrast, settings with longer texts of which different sections have different levels of relevance, embeddings in combination with neural networks will likely outperform the more basic Bag-of-Words models. In both cases, extensive preprocessing can boost the performance. In contrast to applications such as machine translation and categorization of natural texts (such as news articles), BERT models did not outperform the networks using convolutional and recurrent layers. We hypothesize that, as these models require extensive pretraining on external datasets, the technical terminology and telegraphic writing style of the maintenance reports differ too much from the pretraining material to allow for a proper transfer of knowledge to this new domain.

Secondly, we showed how the models can be used to investigate individual labels. If specific labels perform significantly worse, this can be an indication that they are defined suboptimally. For our case study, we have shown there might be room for improvement in the cause labels. Class imbalance can also be investigated. Heavily unbalanced classes can cause issues, because low-support labels are harder to predict. Finally, some applications of the model were illustrated, most notably the error analysis. Because of the high accuracy of the models, only a small fraction of notifications was mislabeled, making it feasible to manually analyse these documents. This investigation provides valuable

insights and can potentially lead to policy improvements. These results are of course highly specific to the case at hand, but the methodology is transferable to related analyses of CMMS data.

While this research has focused specifically on the pharmaceutical industry, there are a plethora of other industries with highly similar CMMS data that could apply the techniques demonstrated in this paper. However, as indicated by our results, there are several interconnected factors that influence the predictive performance (such as input data and label dependence). Therefore, applying the same techniques on new data will require problem specific adaptations.

This work shows that the free text in a failure notification can be used to predict, and potentially correct, failure codes. A problem that was not addressed in the work is how to include SCADA logs into this pipeline. Oftentimes, the sensors attached to a SCADA system are installed with the aim of improving the daily operation of a production facility and its assets and less with the goal of allowing root-cause analysis of a wide range of potential failures in an automated way. Therefore, the data originating from a SCADA system are often only indirectly related to the failure codes. Moreover, for large production plants, the amount of sensor data is huge and sensor readings are affected by a variety of factors (e.g. product-to-product variability, maintenance activities, ambient conditions) not related to failure. Linking our free-text based approach with information derived from a SCADA system is a valuable direction for future work. As a second opportunity, we believe that the development of more technically capable language models, or BERT-type models is an interesting direction for future work. Traditional models are often trained on text corpora that strongly differ from the text that is common in maintenance reports and therefore probably underperform.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[2] Antomarioni, S., Pisacane, O., Potena, D., Bevilacqua, M., Ciarapica, F. E., and Diamantini, C. (2019). A predictive association rule-based maintenance policy to minimize the probability of breakages: application to an oil refinery. *International Journal of Advanced Manufacturing Technology*, 105.

[3] Arif-Uz-Zaman, K., Cholette, M. E., Ma, L., and Karim, A. (2017). Extracting failure time data from industrial maintenance records using text mining. *Advanced Engineering Informatics*, 33:388–396.

[4] Bagadia, K. (2006). *Computerized Maintenance Management Systems Made Easy: How to Evaluate, Select, and Manage CMMS*. McGraw Hill Professional.

[5] Bakliwal, A., Arora, P., Patil, A., and Varma, V. (2011). Towards enhanced opinion classification using nlp techniques. In *Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011)*, pages 101–107.

[6] Cato, W. W. and Mobley, R. K. (2001). *Computer-managed Maintenance Systems: a Step-by-Step Guide to Effective Management of Maintenance, Labor, and Inventory*. Elsevier.

[7] Chalabi, N., Dahane, M., Beldjilali, B., and Neki, A. (2016). Optimisation of preventive maintenance grouping strategy for multi-component series systems: Particle swarm based approach. *Computers & Industrial Engineering*, 102:440–451.

[8] Chen, T., Xu, R., He, Y., and Wang, X. (2017). Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications*, 72:221–230.

[9] Cimini Jr, C. A. and Fonseca, B. Q. A. (2013). Temperature profile of progressive damaged overhead electrical conductors. *International Journal of Electrical Power & Energy Systems*, 49:280–286.

[10] Delobelle, P., Winters, T., and Berendt, B. (2020). Robbert: a dutch roberta-based language model. *CoRR*, abs/2001.06286.

[11] Devaney, M., Ram, A., Qiu, H., and Lee, J. (2005). Preventing failures by mining maintenance logs with case-based reasoning. In *Proceedings of the 59th Meeting of the Society for Machinery Failure Prevention Technology (MFPT-59)*.

[12] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[13] Djatna, T. and Alitu, I. M. (2015). An application of association rule mining in total productive maintenance strategy: An analysis and modelling in wooden door manufacturing industry. *Procedia Manufacturing*, 4:336–343. Industrial Engineering and Service Science 2015, IESS 2015.

[14] Fawcett, C. and Hoos, H. H. (2016). Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, 22(4):431–458.

[15] Honnibal, M. and Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. nl_core_news model.

[16] Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

[17] Krishna, A., Aich, A., Hegde, C., et al. (2018). Analysis of customer opinion using machine learning and nlp techniques. *International Journal of Advanced Studies of Scientific Research*, 3(9).

[18] Li, C., Zhan, G., and Li, Z. (2018). News text classification based on improved bi-lstm-cnn. In *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, pages 890–893. IEEE.

[19] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26.

[20] Mobley, R. K. (2002). *An Introduction to Predictive Maintenance*. Elsevier.

[21] Park, C., Moon, D., Do, N., and Bae, S. M. (2016). A predictive maintenance approach based on real-time internal parameter monitoring. *International Journal of Advanced Manufacturing Technology*, 85(1):623–632.

[22] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

[23] Postma, M., van Miltenburg, E., Segers, R., Schoen, A., and Vossen, P. (2016). Open dutch wordnet. In *Proceedings of the 8th Global WordNet Conference (GWC)*, pages 302–310.

[24] Read, J., Bielza, C., and Larrañaga, P. (2014). Multi-dimensional classification with super-classes. *IEEE Transactions on Knowledge and Data Engineering*, 26(7):1720–1733.

[25] Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2009). Classifier chains for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 254–269. Springer.

[26] Rehurek, R. and Sojka, P. (2011). Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2).

[27] Roberto, S., Fabiana, P., Giuditta, P., and Sergio, C. (2022). Nlp-based insights discovery for industrial asset and service improvement: an analysis of maintenance reports. *IFAC-PapersOnLine*, 55(2):522–527.

[28] Scott, D. and Westcott, V. (1977). Predictive maintenance by ferrography. *Wear*, 44(1):173–182.

[29] Steinberg, D. S. (2000). *Vibration Analysis for Electronic Equipment*. John Wiley & Sons New York.

[30] Suárez, P. J. O., Sagot, B., and Romary, L. (2019). Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Leibniz-Institut für Deutsche Sprache.

[31] Tanguy, L., Tulechki, N., Urieli, A., Hermann, E., and Raynal, C. (2016). Natural language processing for aviation safety reports: From classification to interactive analysis. *Computers in Industry*, 78:80–95.

[32] Trofimovich, J. (2016). Comparison of neural network architectures for sentiment analysis of russian tweets. In *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference Dialogue*, pages 50–59.

[33] Usuga-Cadavid, J. P., Grabot, B., Lamouri, S., and Fortin, A. (2021). Exploring the influence of focal loss on transformer models for imbalanced maintenance data in industry 4.0. *IFAC-PapersOnLine*, 54(1):1023–1028.

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

[35] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[36] Velarde-Suárez, S., Ballesteros-Tajadura, R., and Hurtado-Cruz, J. P. (2006). A predictive maintenance procedure using pressure and acceleration signals from a centrifugal fan. *Applied Acoustics*, 67(1):49–61.

[37] Yamato, Y., Fukumoto, Y., and Kumazaki, H. (2017). Predictive maintenance platform with sound stream analysis in edges. *Journal of Information processing*, 25:317–320.

[38] Yin, W., Kann, K., Yu, M., and Schütze, H. (2017). Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

[39] Zahoor, K., Bawany, N. Z., and Hamid, S. (2020). Sentiment analysis and classification of restaurant reviews using machine learning. In *2020 21st International Arab Conference on Information Technology (ACIT)*, pages 1–6. IEEE.

[40] Zhang, B., Sconyers, C., Byington, C., Patrick, R., Orchard, M. E., and Vachtsevanos, G. (2011). A probabilistic fault detection approach: Application to bearing fault detection. *IEEE Transactions on Industrial Electronics*, 58(5):2011–2018.

[41] Zhang, T., Bhatia, A., Pandya, D., Sahinidis, N. V., Cao, Y., and Flores-Cerrillo, J. (2020). Industrial text analytics for reliability with derivative-free optimization. *Computers & Chemical Engineering*, 135:106763.

[42] Zhang, Z., Robinson, D., and Tepper, J. (2018). Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European Semantic Web Conference*, pages 745–760. Springer.

[43] Zhao, Z., Wang, F.-l., Jia, M.-x., and Wang, S. (2010). Predictive maintenance policy based on process data. *Chemometrics and Intelligent Laboratory Systems*, 103(2):137–143.

[44] Zhong, J.-H., Wong, P. K., and Yang, Z.-X. (2018). Fault diagnosis of rotating machinery based on multiple probabilistic classifiers. *Mechanical Systems and Signal Processing*, 108:99–114.