On the Difficulties of Using Machine Learning on Industrial Texts

Özlem Özgöbek[1], Ole Selvig[1], Jon Atle Gulla[1], Lemei Zhang[1], and Cristina Marco[1]

Department of Computer Science, NTNU, Norway
ozlem.ozgobek@ntnu.no, ole.selvig@gmail.com, jon.atle.gulla@ntnu.no, lemei.zhang@ntu.no, cristina.marco@ntnu.no

Abstract. In oil and gas sector, maintenance reports from a floating oil and gas production, storage and offloading (FPSO) unit play an important role for fixing the failures on time and for learning from previous experiences. Documenting these reports in a structured way is an important task for such units. In this work, we experiment with different machine learning techniques to classify the maintenance reports. However the unstructured text from these reports combined with domain specific terms, abbreviations, misspellings and multiple language use makes the automatic classification of these reports quite challenging. Our findings show that how challenging this task is with the existing report structure and content.

1 Introduction

Offshore oil & gas installations are complex structures that are expensive to build and need continuous monitoring for maintenance and upgrading. It is vital to avoid any unnecessary production stops or accidents, as this would have serious implications for both the companies themselves and their staff. Financially just a short temporary halt of production comes at a substantial cost. More importantly, however, is the risk of serious life-threatening accidents if parts break down or malfunctions arise. Offshore oil & gas production is by nature a challenging task, and the dependence on technology calls for proper procedures for reducing the risk of equipment failures and other accidents that affect the security of people and climate.

Maintenance reports document maintenance activities that are either carried out on a regular basis or triggered by particular events. Whereas some of them address small repairs like the change of broken light bulbs or screws, others concern more time-critical problems that need to be taken very seriously by the management. The intention is to use these maintenance reports to structure the documentation of these activities for later use and facilitate learning and exchange of competence over time. Engineers acquire valuable experiences as they

take part in maintenance activities, but not all of them are equally experienced and no individual engineer can be expected to fully understand the structures and functions of a complete oil & gas installation.

Maintenance reports follow a particular format that forces the engineers to provide some basic information about the maintenance work. There are a number of structured fields that identify for example a piece of equipment, a location or a date, but the engineers are also encouraged to write a short textual description of the work they are carrying out. Even though these descriptions are fairly short and of variable quality, they are crucial when someone wants to check if their present problem has been faced and solved at some time in the past.

Failure codes are of particular interest in maintenance reports. Unfortunately, many reports do not have any failures codes entered, since they were not known at the time the report was created. For later maintenance activities this is problematic for two reasons: (i) A maintenance task is not treated correctly or timely because they fail to realize that similar tasks have been successfully dealt with in the past, and (ii) they are not able to generalize - or learn - from previous cases because a critical piece of data is missing in the documentation of old cases.

In principle, we can approach the missing failure codes as a classification problem. Experiments show, unfortunately, that the structured fields of maintenance reports are not sufficient to produce a classifier with a very high accuracy. The question is whether the textual descriptions may help us fill in the failure codes instead. These texts were not written to support later classification and are probably quite typical for internal documentation in many companies. They were quickly written and with limited knowledge of the problem at hand, and they are best understood by their fellow engineers. The descriptions are neither precise nor complete, but they do reveal something that is potentially relevant for classifying the maintenance work. The contribution of this paper is two-fold: (i) we analyze the challenges of extracting the content of internal industrial texts and assess the usefulness of various linguistic operations, and (ii) we evaluate to what extent machine learning techniques can classify the maintenance reports on the basis of their textual descriptions.

The structure of the paper is as follows. After presenting the dataset in Section 2, we discuss the methods for analyzing the textual descriptions in Section 3. We then use some machine learning techniques to classify the maintenance reports and discuss the results of the experiments at the end.

2 Dataset

The dataset used in this work stems from a floating production, storage and offloading (FPSO) unit in the Norwegian oil and gas industry in the North Sea. It contains a total of 4968 failure reports on various maintenance activities in the oil and gas field. Failure reports are labelled with different code groups according to their association with different parts of the FPSO unit, such as electric generators, steam turbines etc. In the dataset there are 40 code groups. In each code group there are various fault types where each report can be associated

with. However the fault type field that may or may not be filled by the reporter. In the dataset the fault type field is not consistently filled. In general, the reports consist of a number of structured fields and one text describing the failure. In Fig.1 the general structure of the failure reports can be seen.

Report texts describe asset failures from a FPSO unit in the North Sea. The average length of the reports is 75 words, where the minimum length is 7 and the maximum length is 1580 words. An interesting property of the reports in the dataset is that the usage of multiple languages. As it is shown in Fig.2, the languages used in the reports are mostly Norwegian (58.7%) and English (22.6%), however it is possible to see the usage of Danish (3.7%) and Swedish (0.4%) as well. In addition, in 13.2% of the reports multi-language use occurs within the same report. In total there are 32,011 unique words in the reports.

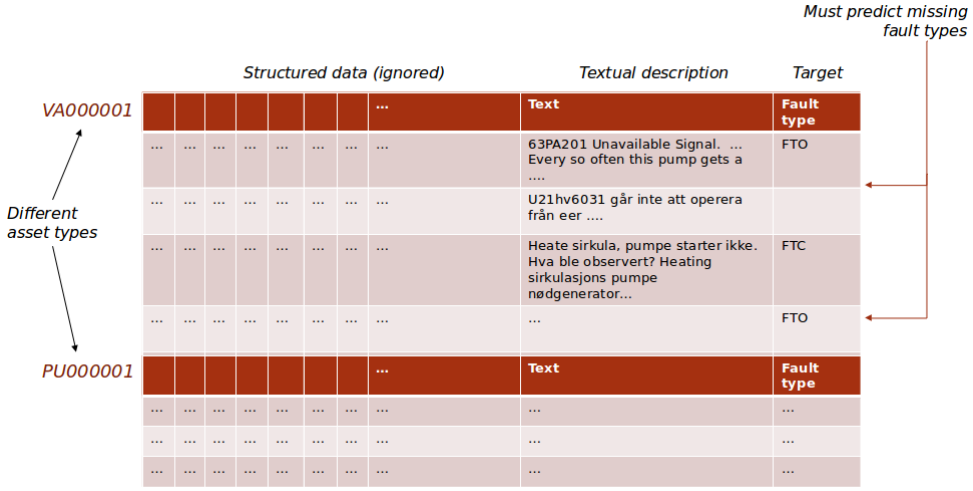

Fig. 1. The general structure and fields of failure reports.

2.1 Challenging properties of the dataset

Dealing with the automatic analysis of the free, unstructured text has challenges. When we consider the technical failure reports at FPSO units, the data comes with even more challenges:

- **Misspellings**: In the failure reports there are several misspelling of words and abbreviations in addition to the usage of incorrect grammar and unfinished sentences.
- **Domain specific terminology and jargon**: The usage of domain specific terms and acronyms is available in the failure reports.

	Events	Percent
Swedish	19	0.4
Danish	185	3.7
Multi/Bi	657	13.2
Norwegian Nynorsk	67	1.3
Norwegian	2915	58.7
English	1123	22.6

Fig. 2. Different languages used in maintenance reports.

- **Multi-language use**: As mentioned earlier, 13.2% of the failure reports has the usage of multiple languages at once.
- **Abbreviations**: For the ease of reporting there are several abbreviations available in the reports. Several of those abbreviations are not explained and it is only possible for the human domain experts to understand.
- **Non-standard structures**: The structure of reports is not standardized. Even though it seems that there is a form of structure in some of the reports, many of the reports do not follow the same structure.
- **Imbalance in report types**: The distribution of 4968 failure reports among 40 different code groups is not balanced. As it is shown in Fig. 3 the difference between the distribution of reports vary a lot.

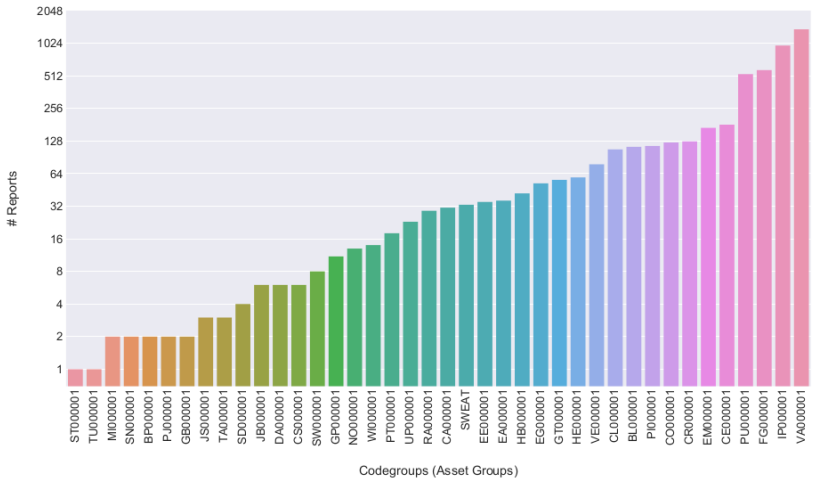

Fig. 3. Imbalance in report types.

3 Methods

3.1 Text simplification and pre-processing

Prior to conduct the classification experiments we conducted some normalization of the texts. This normalization includes some text simplification strategies as well as general NLP pre-processing techniques.

Sentence boundaries were standardized as much as possible, in order to enhance the performance of the classifiers. In particular, newlines and non-standard sentence boundaries were normalized by substituting the symbols with a basic dot. Regular expressions were used to do these transformations.

Besides, most reports include specific dates and times describing when the report was made or edited and by whom. This information was also transformed into general named entities labels so as to reduce the vocabulary space the classifier has to deal with as much as possible. Figure 4 shows an example of an original maintenance report and Figure 5 illustrates the same text after simplification. This example illustrates how actual dates, times and proper names for equipment, location, people and measurement numbers, were mapped into general labels during normalization. Protocol questions included in most reports as templates for employees were also removed, with the purpose of excluding unnecessary information.

```
Bytte manometer og testventil BB
28.10.2015 15:53:00 OLA NORDMANN (ONRODMANN)
-Hva ble observert?
Defekt manometer og testventil
-Lokasjon for feil?
Aux rom babord side
-Hva har du allerede utført?
Prøvd og finne deler, men vi må bestille
-Har vi sett denne feilen før?
Ja
-Hva må til for å reparere denne?
(Forslag til linjer for arbeidsordre)
Bestille deler og bytte
-Hvilke Sikkerhets/Miljø/Produksjon påvirkning har denne feil?
Vi mister avlesning ved brannpumpetest
12.12.2016 Deler till denne jobben ligger på WO 200023249
******************************
30.01.2016 KARI
Deler ligger i TU-2-A
******************************
```

Fig. 4. Maintenance report example. Original text.

```
Bytte manometer og testventil BB DATE TIME NAME NAME (NAME) Defekt manometer og
testventil Aux rom babord side Prøvd og finne deler, men vi må bestille Ja Bestille
deler og bytte Vi mister avlesning ved brannpumpetest DATE Deler till denne jobben
ligger på WO TAGNUM DATE NAME Deler ligger i TAGNUM
```

Fig. 5. Maintenance report example, after text simplification.

After these simplification, standard NLP pre-processing was performed, including tokenization, normalization of words with capital letters, stop-words removal and stemming, whereby words are reduced into their base form. NLTK was used for these purposes.

3.2 Feature extraction

Two different feature models were explored in this study. A baseline model, consisting on a bag-of-words model including TF-IDF frequencies, and two word embedding models, in particular distributed memory (DM) and distributed bag of words (DBOW) were explored. Gensim and scikit-learn were correspondingly used for these purposes.

The approach for obtaining such features varies slightly. TF-IDF models are fitted individually with text data from each code group or class. In contrast, document embedding models are fitted with the text data from all classes, and then each of the selected class infers embeddings from the same global model.

3.3 Classification

Text classification is a fundamental field that has been well studied by researchers and practitioners of a variety of fields, including artificial intelligence, statistics, pattern recognition, cognitive psychology, computer vision and medicine. In this section, we first formally define the classification task. Then different types of classification algorithms, specifically, K nearest neighbor, Multinomial Naive Bayes, Random Forest and Support Vector Machine, are adopted and explained.

Problem definition Let $\mathcal{X} = \{\mathbf{x_1}, \mathbf{x_2}, , \mathbf{x_N}\}$ be the input set after pre-processing, and y be the output being assigned to a class $c \in \mathcal{C}$ after generated by inference model f. $\mathbf{x_i}$ is the input vector of the i-th document, and N represents the number of documents in training set. The classification problem can be denoted as: $y = f(\mathcal{X})$. Our goal is to assign y to a class c which is the same with ground truth class c.

K Nearest Neighbor The fundamental mechanism behind nearest neighbor method is to find K training samples which are closest in distance to the newly arrived input point [4]. The distance can be any types of metric measurements according to specific applications. In this paper, euclidean distance is adopted to find the closest K points in training set.

$$d(\mathbf{x}, \mathbf{x}) = ||(\mathbf{x} - \mathbf{x})|| = \sqrt{\sum_{\mathbf{i}}^{\mathbf{D}} (\mathbf{x_i^2} - \mathbf{x_i}^{\mathbf{2}})} \tag{1}$$

https://www.nltk.org/
https://radimrehurek.com/gensim/
https://scikit-learn.org/

where $\mathbf{x}$ and $\mathbf{x}$ are the representations of training and target points respectively. $d(\cdot)$ is the distance between $\mathbf{x}$ and $\mathbf{x}$ and D is the dimension of input vector. The predicted class y can be achieved according to the conditional probability given the set of the nearest K neighbors $\mathcal{Z}$:

$$\hat{y} = argmac_{c \in \mathcal{C}} p(y = c|Z) = \frac{1}{k} \sum_{i \in \mathcal{Z}} I(y_i = y) \tag{2}$$

where $I(\cdot)$ is the indicator function having the value of 1 if y belongs to a specific class else 0.

Naive Bayes Naive Bayes classifier classifies newly input documents using Bayes rule under the assumption of the independence between input features [7]:

$$\hat{y} = argmac_{c \in \mathcal{C}} p(c) \prod_{i=1}^{D} p(x_i|c) \tag{3}$$

Multiple models can be used to calculate the posterior probability of the relevance of input $\mathbf{x}$ given class c, among which includes Multinomial [6] and Guassian models [8]. In our case, Multinomial model is used since it can cover both discreet and continuous feature spaces [9].

$$p(\mathbf{x}|\mathbf{c}) = \frac{(\sum_{\mathbf{i}} \mathbf{x_i})!}{\prod_{\mathbf{i}} \mathbf{x_i}!} \prod_{\mathbf{i}} \mathbf{p}_{\mathbf{c_i}}^{\mathbf{x_i}} \tag{4}$$

where x_i is the relative value of the i-th feature of input document.

Random Forest Random Forest is an ensemble of decision tree algorithms [2], where each decision tree participates in the class prediction process through result aggregation. Decision trees embody the classification approach [14] through a tree-like structure, which is made up of a root and internal nodes, branches and leaves. Branches represent paths denoted by a range of values and internal nodes control splitting rules represented by selected features from input space. Information Gain is one of the most popular feature selection methods adopted in this paper at each internal node during splitting process.

$$InformationGain(\mathcal{X}_p, x_i) = H(\mathcal{X}_p) - \sum_{j=1}^{m} \frac{N_i}{N_p} H(\mathcal{X}_j) \tag{5}$$

where $\mathcal{X}_p$ is the set of training documents before split, and $\mathcal{X}_j \in \mathcal{X}_p$ is a subset of documents disjoint with each other after split. m is the number of values of feature x_i appeared in training set. N_i and N_p are the number of documents in $\mathcal{X}_p$ and $\mathcal{X}_j$ respectively. $H(\cdot)$ specifies the entropy. The feature and value are selected with the highest Information Gain value recursively until the splitting process reaches the leaf node and is assigned to a class. Thus, the final classification result of Random Forests is given as the averaged prediction of the individual decision trees.

Support Vector Machine Support Vector machine (SVM) is another method that we use to categorize fault type labels of text descriptions [5]. SVM sets out to fit a hyperplane with largest margin to nearest examples of any class. These examples are known as support vectors. The optimization process of SVM is to support vectors with largest margin between arbitrary classes while minimizing the lost function L. In our paper, the extension version of hinge loss [13] is adopted to achieve the best margin. Specifically, for

$$L = \sum_{j \neq c} max(0, \mathbf{W_j^T x} - \mathbf{W_c^T x} + \delta) + \lambda \cdot \mathcal{R}(\theta)$$

where $\mathbf{W_j}$ and $\mathbf{W_c}$ are model parameters. δ is the tolerance threshold to be tuned during training process. $\mathcal{R}(\cdot)$ represents the regularization term used to prevent over-fitting problems and model complexity, which can be either L_1 or L_2 regularization. θ represents the set of model parameters and λ is the regularization parameter.

4 Experiments

Different experiments run in different phases: 1- NLP techniques in preprocessing 2- Machine learning algorithms for classification

4.1 Evaluation Metrics on Classifications

Here we enumerate the evaluation metrics on classification tasks adopted in this paper. The formulae are shown in Table 1.

- **Accuracy (ACC)**: Despite its popular property in evaluation of classification tasks, some issues occur in an imbalanced domain. For instance, consider a problem where only 1% of the examples belong to the positive class, a accuracy of 99% is achievable by predicting negative class for all examples. Thus, ACC is at best misleading, when the ratio between positive and negative examples is significantly skewed.
- **Harmonic Mean of Precision & Recall (F_β)**:The definition is based on two metrics, recall and precision. Recall captures the completeness of how many positive examples are predicted positive, while precision captures the exactness of how many predicted positive examples are true positive. Because of an imbalanced multi-class data distribution in our experiments, we adopt macro averaging in [10] to summarize the classifiers performance over a set of classes $\mathcal{C}$. Thus, the example size of a class does not affect the measurement. For a given class c, F_β is defined as the harmonic mean of precision and recall [12]. β is introduced to adjust the relative importance of recall with respect to precision and is set to 1 in our case.
- **Cohens Kappa (κ)**: It compares an observed accuracy with and expected accuracy [3]. Specifically, κ provides a score ranging from -1 to 1, from which the performance of a classifier can be evaluated with respect to random

chance. The bigger the value is, the better the performance shows. The interpretation of κ score follows the work in [1].
- **Geometric Mean Accuracy (GMA)**: The metric takes the geometric average (GMA) of the recall score in each class and it benefit from the use of geometric mean when there is a multiplicative or exponential relationship between class scores averaged especially for imbalanced data distribution [11].

Besides, two guessing strategies are chosen as baselines:

- **Stratified (STRAT)**: It generates predictions by respecting training sets class distribution.
- **Most Frequent (MFREQ)**: It always predicts the most frequent label in the training set.

Table 1. Classification evaluation metrics

Evaluation Metrics	Definition
Accuracy (ACC)	$ACC = \frac{tp + tn}{tp + tn + fp + fn}$
F_β	$F_\beta = \frac{(1+\beta)^2 \cdot Prec_M \cdot Rec_M}{\beta^2 \cdot Prec_M \cdot Rec_M}$
Cohens Kappa (κ)	$\kappa = 1 - \frac{1 - p_o}{1 - p_e}$
Geometric Mean Accuracy (GMA)	$GMA = \sqrt[\lvert\mathcal{C}\rvert]{\prod_{c \in \mathcal{C}} recall(c)}$

4.2 Parameter Settings

Hyper parameters are selected through grid-search for different tasks and classifiers. Then values with best performance of classification tasks are selected. Each sampled experiment for a particular feature extraction method will be scored in terms of its exhibited F_β macro averaged score with $K = 5$ in K Nearest Neighbor predictions.

---

where tp, tn, fp and fn denote true positive, true negative, false positive and false negative respectively

$Prec_M = \frac{\sum_{c \in \mathcal{C}} precision(c)}{|\mathcal{C}|}$ where $precision(c) = \frac{\sum_i I(y_i = c) I(\hat{y} = c)}{\sum_i I(\hat{y} = c)}$. $Rec_M = \frac{\sum_{c \in C} recall(c)}{|\mathcal{C}|}$ where $recall(c) = \frac{\sum_i I(y_i = c) I(\hat{y}_i = c)}{\sum_i I(y_i = c)}$

where p_o denotes the observed accuracy and p_e denotes the expected accuracy of classification performance

5 Evaluation

As described before, there are several properties in oil and gas maintenance reports which make them particularly challenging to be automatically understood: specialized terminology and expressions, multiple languages, misspelled and non-standard words, inconsistent syntax and lack of proper sentence boundaries.

In order to assess the impact of the different simplification and pre-processing techniques, as well as feature extraction models, a careful evaluation was performed. This evaluation was implemented in three steps: (i) assessment of the feature extraction methods; (ii) evaluation of singular simplification and pre-processing techniques; (iii) evaluation of simplification and pre-processing methods in combination.

5.1 Assessment of the feature extraction methods

As results from experiments with each of the feature extraction methods show, each method performs differently on different models. Figures 6, 7 and 8 show the distributional graphs containing all the models sampled during hyperparameter search. In these figures, the vertical axis represents the total number of models that achieved any of the F scores along the horizontal axis. Note that axis scales are not consistent among the graphs.

The graphs show that the performance of document embeddings (DM and DBOW) varies significantly. Although TF-IDF show a more consistent performance across various hyperparameters, and exhibited the best performance of all the three methods, it also includes the worst performing model.

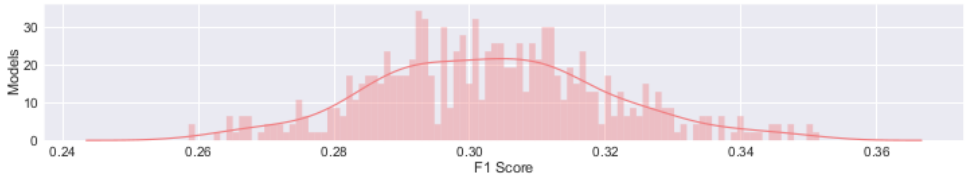

Fig. 6. Distributed memory (DM)

5.2 Evaluation of individual text simplification and pre-processing techniques

Table 9 shows classifier performance for the text simplification techniques explored here. As can be seen from this table, both SVM and MNB perform below a simple baseline, questioning the validity of using text simplification techniques for the classification of maintenance reports. Specially it is low in all cases where

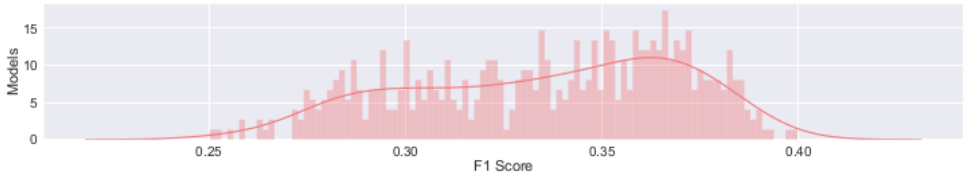

Fig. 7. Distributed bag of words (DBOW)

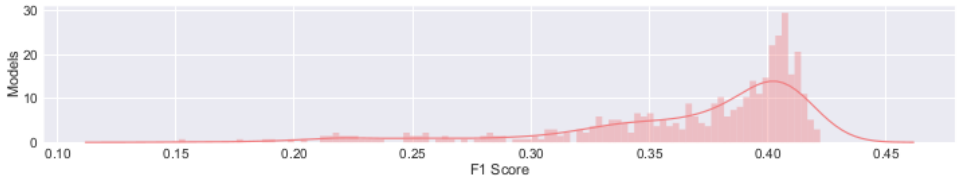

Fig. 8. Term Frequency-Inverse Document Frequency (TF-IDF)

Experiment	F_β	CLF	GMA	ACC	κ
question-structure filtering	0.419	MNB	0.521	0.518	0.452
	0.381	SVM	0.468	0.507	0.440
	0.358	RF	0.494	0.507	0.432
baseline	0.406	MNB	0.509	0.504	0.436
	0.402	SVM	0.412	0.511	0.445
	0.353	RF	0.400	0.489	0.412
person-name filtering	0.379	MNB	0.467	0.478	0.404
	0.365	SVM	0.539	0.504	0.437
	0.339	RF	0.467	0.496	0.421
datetime-number-tag filtering	0.409	MNB	0.511	0.504	0.435
	0.403	SVM	0.420	0.504	0.439
	0.361	RF	0.500	0.515	0.440
datetime filtering	0.421	SVM	0.455	0.518	0.455
	0.401	MNB	0.504	0.496	0.427
	0.339	RF	0.469	0.489	0.410

Fig. 9. Classifier performance by text simplification technique. Classifiers in each group are ranked in terms of F score exhibited with each technique.

the performance of classifiers with person-name filtering. Also it is possible to see that different classifiers perform better on different experiments.

Table 10 shows the results of the classifiers when performing standard NLP pre-processing. Interestingly, the SVM experiment including stemming yielded the best results, while the worst results were obtained for the MNB classifier under the same conditions. The reverse can be seen with non-alphanumeric filtering. In general, these pre-processing methods had a much a bigger impact on the performance of SVM, yielding results above the baseline. In contrast, MNB got results over the baseline only when non-alphanumeric filtering was performed.

Experiment	F_β	CLF	GMA	ACC	κ
baseline	0.413	MNB	0.513	0.518	0.452
	0.386	SVM	0.493	0.493	0.425
	0.353	RF	0.402	0.493	0.416
stemming	0.439	SVM	0.524	0.562	0.502
	0.397	RF	0.370	0.518	0.443
	0.385	MNB	0.483	0.489	0.420
stop words filtering	0.435	SVM	0.446	0.547	0.484
	0.412	MNB	0.512	0.518	0.453
	0.350	RF	0.482	0.500	0.422
non-alphanumeric filtering	0.414	MNB	0.516	0.518	0.452
	0.390	SVM	0.477	0.515	0.449
	0.345	RF	0.480	0.496	0.420
lowercase	0.414	SVM	0.423	0.515	0.449
	0.409	MNB	0.497	0.511	0.443
	0.350	RF	0.400	0.471	0.389

Fig. 10. Classifier performance by NLP pre-processing technique. Classifiers in each group are ranked in terms of F score exhibited with each technique.

5.3 Evaluation of individual text simplification and pre-processing techniques in combination

Lastly, we evaluated the performance of the different pre-processing methods in combination. The top five performing combinations of pre-processing methods for SVM and MNB can be seen in Tables 11 and 12.

qs: question structure filtering; sw: stop-words filtering; l: lowercase (no capitalization letters); st (stemming); dnt: datetime-number-tag filtering; nlta: non-alphanumeric filtering; dt: datetime filtering

Experiment	CLF	F_β	GMA	ACC	κ
qs, l	SVM	0.453	0.483	0.551	0.494
dnt, nlta, l, st, sw	SVM	0.453	0.408	0.536	0.477
dnt, nlta, st, sw	SVM	0.453	0.408	0.536	0.477
qs, sw	SVM	0.450	0.482	0.529	0.468
nlta, l	SVM	0.448	0.460	0.566	0.509

Fig. 11. SVM: The top 5 best performing pre-processing and simplification techniques in combination, ranked by F score.

Experiment	CLF	F_β	GMA	ACC	κ
d, nlta, l, sw	MNB	0.439	0.453	0.540	0.478
nlta, l, sw	MNB	0.441	0.453	0.540	0.477
l, sw	MNB	0.426	0.439	0.526	0.461
d, l, sw	MNB	0.425	0.439	0.526	0.461
qs, nlta, l	MNB	0.421	0.516	0.533	0.469

Fig. 12. MNB: The top 5 best performing pre-processing and simplification techniques in combination, ranked by F score.

Although there is no consistent ranking between the two classifiers, however, lowercase, stop words filtering and non-alphanumeric filtering appear amongst the techniques with most impact on their performance. This is not surprising, as these techniques are generally used as pre-processing steps in most NLP pipelines. Slightly more unusual is the fact that stemming does not seem to be very relevant for the classification algorithm's performance. We suggest that this might be due to the fact that there is a high number of spelling errors and specialized vocabulary which can make the vocabulary space simplification more challenging.

6 Discussion and Conclusion

In this work we looked into the maintenance reports from a floating production, storage and offloading (FPSO) unit in the Norwegian oil and gas industry. These reports are used to structure the documentation of these activities for later use and facilitate learning and exchange of competence over time.

Failure codes which some of the reports in the dataset include, help the engineers to classify the type of failures in order to treat the failures on time and to learn from the similar failures happened in the past. However, not all the reports include failure codes. In this work, we use machine learning in order to predict the missing failure codes. By using the textual description of the failure, after pre-processing of the text and feature extraction, we experimented with three different machine learning classifiers: K Nearest Neighbor, Naive Bayes and Support Vector Machine. As explained in the previous sections, the results show that different classifiers outperforms other methods on different tasks. But unfortunately none of the methods are good enough to be useful in this task and tested methods do not solve challenges. The challenges of the dataset, as explained in detail in Section 2.1, makes data preparation and analysis very hard. Maintenance reports have special characteristics. Usage of multiple languages at once, unstructured text with misspellings, technical terms, abbreviations are quite challenging for the existing natural language processing methods to deal with.

References

1. DG Altman. Practical statistics for medical research chapman & hall london google scholar. *Haung, et al [16] USA (Black)*, 1991.
2. Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
3. Jean Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational linguistics*, 22(2):249–254, 1996.
4. Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.
5. Mark M Kornfein and Helena Goldfarb. A comparison of classification techniques for technical text passages. In *World congress on engineering*, pages 1072–1075, 2007.

6. David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc., 1994.
7. Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.
8. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.
9. Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 616–623, 2003.
10. Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.
11. Yanmin Sun, Mohamed S Kamel, and Yang Wang. Boosting for learning multiple classes with imbalanced class distribution. In *null*, pages 592–602. IEEE, 2006.
12. CJ Van Rijsbergen. Information retrieval. dept. of computer science, university of glasgow. *URL: citeseer. ist. psu. edu/vanrijsbergen79information. html*, 14, 1979.
13. Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *Esann*, volume 99, pages 219–224, 1999.
14. Yongheng Zhao and Yanxia Zhang. Comparison of decision tree methods for finding active objects. *Advances in Space Research*, 41(12):1955–1959, 2008.