Leveraging Free-form Text in Maintenance Logs Through BERT Transfer Learning

Syed Meesam Raza Naqvi[1,2], Christophe Varnier[1], Jean-Marc Nicod[1], Noureddine Zerhouni[1], and Mohammad Ghufran[2]

[1]*FEMTO-ST institute, Université de Bourgogne Franche-Comté / CNRS / ENSMM, Besançon, France*
[2]*Fives CortX, Lyon, France*
{*syedmeesam.naqvi, christophe.varnier, jm.nicod, noureddine.zerhouni*}*@femto-st.fr*
{*meesam.naqvi, mohammad.ghufran*}*@fivesgroup.com*

Abstract

Across various industries, maintenance entries recorded over time contain decades of experience and health records of different assets. Maintenance logs are usually free-form text entries recorded by maintenance operators. These records are also highly unstructured and imbalanced. Because of these reasons, this huge source of knowledge is usually underutilized and does not contribute to the development of tools to help improve maintenance processes. In the last few years, due to recent advancements in the field of Natural Language Processing (NLP) and increased focus on industry 4.0, there is a need to revisit this problem. This study explores the use of state-of-the-art NLP methods on free-form maintenance text to leverage this data. More specifically, the purpose of the study is to estimate the problem category of maintenance log to see how well recent NLP models adapt to free-form maintenance text. Findings of this study indicate that CamemBERT with the fine-tuning approach outperforms the classical NLP approaches. Class imbalance problem is also addressed through data augmentation using deep contextualized embedding showing further performance improvement. Model accuracy and Matthews correlation coefficient (MCC) are used as performance metrics to give a better understanding of results with imbalanced classes.

Keywords— Industry 4.0 (I4.0), NLP, BERT, Maintenance logs, Classification.

1 Introduction

As we are moving forward into the era of Industry 4.0 (I4.0), data has proved to be the key player in the industrial revolution. Data-driven Artificial Intelligence (AI) powered solutions are transforming industrial processes. Due to these advancements in the field of AI and big data analytics, data is now called 'the new oil' [6]. Across various industries, documents recorded over time contain decades of knowledge obtained through experience. It also contains valuable health records and maintenance histories. The main problem is that the information in these documents is technical and is mostly free-form (unstructured) text. Natural language processing (NLP) is a branch of computer science that deals with the application of computational methods for the automatic analysis and representation of human language [11]. Contrary to regular languages, in maintenance logs, both vocabulary and the document structure are significantly different from data used in training general NLP pipelines. As a result, this information cannot be readily processed using these pipelines. To address and formalize this issue, General Electric (GE) Digital, in collaboration with researchers from the National Institute of Standards and Technology (NIST) and the University of Western Australia, recently published a study. In this study, they proposed Technical Language Processing (TLP) as a subdomain of AI. TLP can be considered as a way to adopt NLP pipelines for industrial use cases to develop reproducible and scalable industrial solutions [3]. Across industries, Maintenance Work Orders (MWO) are among common sources of technical language data. MWO planning, scheduling, and execution generate a significant amount of data over the life cycle of assets. These data, collected over the years, contain

information about asset performance, maintenance policies, and failure patterns. If efficiently utilized, this information can be converted into actionable intelligence. However, the information in maintenance records is usually underutilized because 1. vocabulary and sentence structures vary significantly from natural language texts, 2. Information is not structured and not uniformly presented, and 3. Lack of good quality data. Recent trends in NLP have minimized the dependency on these classical preprocessing pipelines. In this study, we will explore the capability of state-of-the-art techniques in NLP on unstructured maintenance text. To process text using a Machine Learning (ML) algorithm, the first step is to convert it into numerical representation through vectorization. One of the widely used classical approaches for vectorization of textual information is Term Frequency-Inverse Document Frequency (TF-IDF). Given a set of documents, the idea behind TF-IDF is to convert each document into a vector by giving more importance to the terms that are unique to a given document compared to common terms in all available documents. In TF-IDF, frequency of each term is multiplied by a weight that depends on the rarity of that term in the common vocabulary of all documents [7]. TF-IDF is one of the most widely used techniques in the field of NLP [1].

In 2013, there was a shift in NLP research with the introduction of non-contextualized word embedding, a vector representation of words. Techniques like Word2Vec by Google [12], GloVe by Stanford [13] and FastText by Facebook [2] transformed classical preprocessing pipelines. FastText can divide a word into subforms, thus automatically handling misspelled words and abbreviations. In this way, FastText converts the words often ignored by classical preprocessing pipelines into known entities. Such features help in processing free-form text as it contains abbreviations and short forms of words. These non-contextualized embedding-based vectors preserve the semantic meaning with a certain degree of underlying language understanding. The main disadvantage of non-contextualized word embedding is that they are static, meaning each word has a fixed vector representation regardless of the context. For instance, resulting vectors can capture relations between words, such as vector for "king" – "man" + "woman" results into vector close to "queen" (gender relationship). The vector for "Paris" – "France" + "Poland" results in a vector close to "Warsaw" (capital relationship). But for polysemic words such as "bank", when used as river bank and financial bank, will have the same fixed embedding regardless of the context it is used in. This drawback led the research towards the development of techniques to generate contextualized word embedding using techniques like recurrent neural networks [10] [14] and attention mechanisms [20] [4]. These efforts led to the development of famous NLP models like ELMo [15], OpenAI GPT [16] and BERT [4]. BERT introduced a novel training methodology based on transformers architecture that takes advantage of the attention mechanism presented in [20]. This bidirectional learning approach achieved very good performance, pushing state-of-the-art on 11 downstream tasks including classification, language inference, Named Entity Recognition, question-answering, and others. After the release of BERT as an open-source technique, different NLP groups were able to produce similar state-of-the-art results on different languages and tasks [21]. One of the main advantages behind the success of BERT is transfer learning - the ability to adopt large pre-trained deep learning models across different languages, tasks and domains [17]. This feature of BERT accelerated progress in the field of NLP that ImageNet brought to the computer vision community.

This study focuses on predicting problem categories in maintenance records written in French language. We used the French version of BERT (CamemBERT) which is based on RoBERTa's architecture and achieved state-of-the-art results on various French NLP tasks. This model is trained on 138GB of uncompressed French text through 17 hours of GPU training [9]. Literature analysis shows that the use of contextualized word embedding is not very common on free-form text especially under the context of Industry 4.0. To the best of the author's knowledge, this is the second time CamemBERT is used on free-form industrial dataset, with the first study being [19]. The remainder of this paper is organized as follows: Section 2 will discuss the overall methodology : data preprocessing and augmentation steps, and ML architectures used to develop the classification system. Section 3 will describe the results in detail and Section 5 will present the directions for future work. Finally, paper is concluded in the Section 4.

2 Methodology

This section presents the dataset, along with a description of preprocessing steps. Also, the architecture and methodology used to develop the classification system are explained.

Figure 1: Sample maintenance record before and after preprocessing

Input before preprocessing

Fault description	Symptoms	Category
Description: Abnormal noise at carousel and power failure on the lower carousel ***Diagnosis:*** Motor out of service ***Cause:*** Broken bearing	Anormal noise	Mechanical

Input after preprocessing

Text	Label
abnormal noise at carousel and power failure on the lower carousel. motor out of service. broken bearing. abnormal noise.	Mechanical

Table 1: Description of dataset used for study

Attribute	Data type	Details
Fault description	String *(free-form text)*	Description of fault, diagnostic steps involved and cause of the fault
Symptoms	String *(free-form text)*	Description of symptoms of the fault
Category	String *(categorical)*	Four problem categories: 1. Mechanical 2. Electronic 3. Electrical 4. Miscellaneous

2.1 Dataset preprocessing

Dataset used in this study is obtained from a manufacturing company. Due to confidentiality constraints, the name and industry are not mentioned in this paper. The dataset contains maintenance records collected over the years from 2015 to 2021. These records contain a total of 814 entries classified into four problem categories: *mechanical*, *electronic*, *electrical* and *miscellaneous*. The miscellaneous category consists of samples from three different categories (information technology, network, and pneumatic) combined into a single category due to lower sample counts. The details of the available variables are described in Table 1 and Figure 1 shows a sample of the maintenance records before and after preprocessing.

One of the main reasons behind the selection of deep contextualized embedding is the minimal preprocessing requirement. This saves a lot of time designing preprocessing pipelines using different NLP toolkits. BERT can process the text in its raw form. In our case, the fault description variable was in two different formats: it either contained the problem description without tags or problem description with tags (description further divided into the description, diagnostic, and cause). As most of the problem descriptions are without tags it was decided to normalize all the descriptions into the same format: as description without tags. To do that, tags were removed to make a single continuous problem description. Another preprocessing step performed is the concatenation of symptoms at the end of each description to generate a single input sequence for each maintenance record. The concatenation of symptoms with description was decided due to the high percentage of missing values in symptoms.

2.2 Dataset distribution, augmentation and splitting

In Table 2 it can be observed that there is an imbalance between the sample count of different classes. To overcome this issue data augmentation is applied.

Table 2: Distribution of samples among different problem categories

	Mechanical	Electronic	Electronic	Miscellaneous	**Total**
No. of samples	308	234	143	129	**814**

Data augmentation has been widely used in computer vision but it is also gaining popularity in the NLP community for studies where resource data is low or class imbalance poses difficulties in training and

Figure 2: Flow diagram of data splitting and augmentation

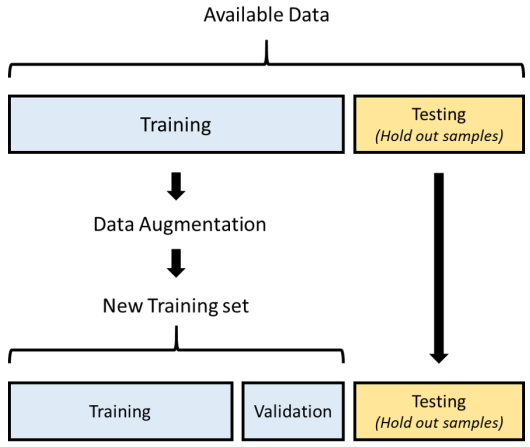

Table 3: Distribution of training and test samples with and without augmentation

Category	**Training samples**		**Test samples**
	without augmentation	*with augmentation*	*Holdout samples*
Mechanical	277	277	31
Electronic	208	277	26
Electrical	133	277	10
Miscellaneous	114	277	15
Total	**732**	**1108**	**82**

evaluation phases.

Another reason behind the popularity of data augmentation in the NLP community is the introduction of very large-scale transformer-based neural network models that require large amounts of data for training [5]. After preprocessing step, to avoid the introduction of bias in the model, data is split into train and test sets before augmentation. Figure 2 shows the flow diagram of data splitting and augmentation. A train-test split of 90%-10% is used in order to maximize the amount of training data. This is done because deep contextualized embedding models, like BERT, need large quantities of data.

To obtain a balanced training set and increase training data, data augmentation was applied using the *nlpaug* Python package [8]. This package provides different augmentation strategies including character, word, and sentence level augmentation. For each of these strategies, augmentation is performed through random substitution, random deletion, random insertion, shuffling, and synonym replacement, etc. *nlpaug* package also supports augmentation using classical techniques like TF-IDF, Word2vec and also has the capability of augmentation using more advanced NLP techniques such as BERT and XLnet based contextualized word embedding. As our study is focused on the role of contextualized embedding based on BERT, we also used this technique for augmentation. Using CamemBERT-large masked language model, a random substitution was performed using contextualized word embedding. As augmentation introduces some unwanted noise and there is a chance that the new data is very different from the actual data it was decided to keep the augmentation to a minimum. Data augmentation was therefore only applied to less frequent classes (electronic, electrical, and miscellaneous) compared to the most frequent class (mechanical).

To augment each class with an equal number of samples, training data is divided into separate classes (mechanical, electronic, electrical, and miscellaneous). Then for every class (except mechanical), each data sample is used to generate five augmented samples. Finally, the required number of randomly selected augmented samples were added to original training examples to generate a new training set with an equal number of samples (277 samples) per class. Table 3 shows the distribution of samples among training and test with and without augmentation. To avoid bias in the model, the test set is kept the same as the one obtained through data splitting before augmentation. After creating the new training set, different ML models were trained using various feature combinations. In the next section different ML models trained to develop the classification system are discussed in detail.

2.3 Machine Learning Models

To test the performance of various classification techniques, models were trained using five different approaches. To see the performance gain due to data augmentation, separate models were trained with and

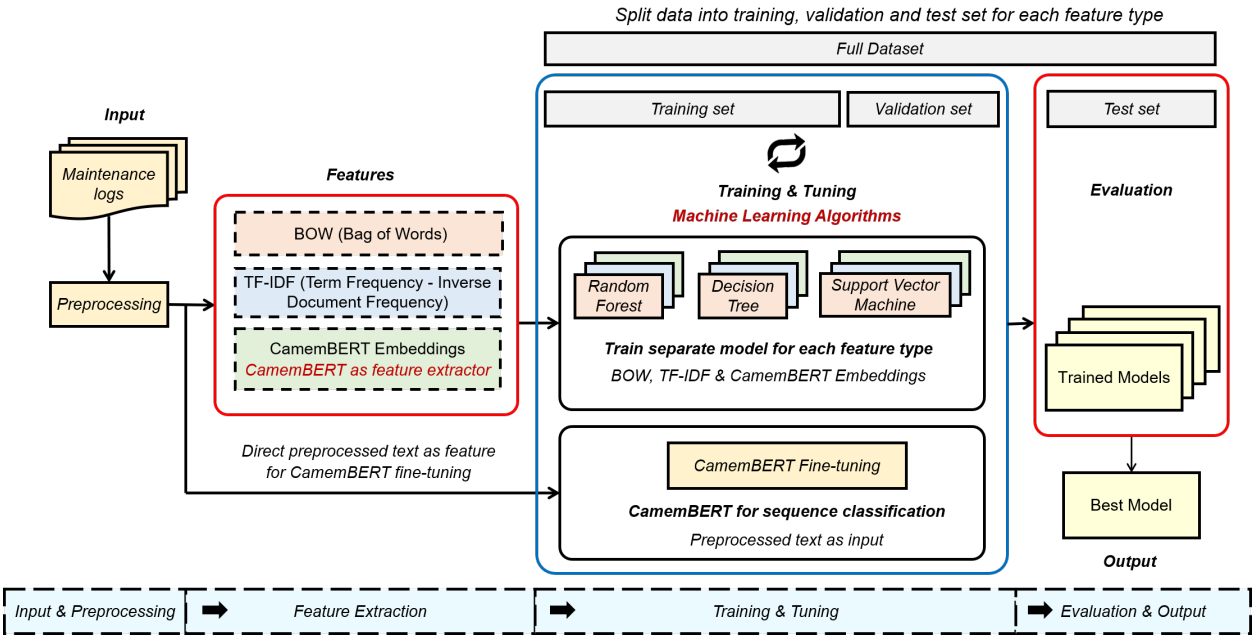

Figure 3: Model development pipeline from input to output (best model) - *same pipeline is followed for training data with and without augmentation.*

without augmentation. In addition, to show the effectiveness of deep contextualized embedding on free-form text, deep contextualized embedding-based models are compared with models trained using classical features such as bag of words (BOW) and TF-IDF. TF-IDF is one of the most used methodologies for text vectorization in NLP [1]. Figure 3 shows an overview of model development pipeline from input to the selection of best model, including different types of features and models used in the study. Different approaches used are discussed in detail in the following subsections.

2.3.1 Baseline model:

To test the performance of all models with a baseline, a dummy model was developed. This model only serves as the baseline against other models and can not be used as the final model for classification. The approach used to develop the dummy model was predicting the most frequent class that is *mechanical* in our case.

2.3.2 Hyperparameter Tuning:

The performance of a machine learning model significantly depends on the chosen hyperparameters. Manual hyperparameter tuning is time-consuming and it is difficult to try so many combinations. To avoid manual tuning and automate this process, grid search for hyperparameters was applied. To use grid search we used *GridSearchCV* function of *sklearn's model_selection package*. Using this function, we can identify the best set of hyperparameters to fit the model on the training set. To find the best set of hyperparameters we provide *GridSearchCV* function with a dictionary of hyperparameters containing hyperparameters and possible values the hyperparameters can take. Then *GridSearchCV* function tries all the possible combinations of these parameters and evaluates the model for each combination using cross-validation. In the end, the best parameters can be selected for the final model by comparing the performance metrics. Only the hyperparameters in the provided dictionary are tuned and the default value of the rest of the hyperparameters are used. Table 4 shows the hyperparameters along with provided value ranges for the different models used in the grid search. Hyperparameter tuning of CamemBERT fine-tuning-based models is discussed in Section 2.3.5.

2.3.3 Bag of Words feature based models:

To convert training data into classifier input, each sample needs to be converted into a numeric vector. In the BOW approach, a vocabulary is generated that is a list of all the unique words in the data set. Using this vocabulary, a fixed numerical vector is created for each training sample, using the count of each unique word in the given training example. To reduce the size of vocabulary and vectors, stop words were removed from the data before vectorization. After generating features for both training sets (with and without augmentation), three different types of models are trained using these features: Support Vector Machine (SVM), Decision Tree (DT), and Random Forest (RF). The best models for these models types are chosen based on the best performing hyperparameters using the grid search parameters presented in Table 4.

2.3.4 TF-IDF feature based models:

TF-IDF is the product of two measures i.e. term frequency, and inverse document frequency. In the first step term frequency (TF) is calculated. Term frequency, for a given document measures the number of times a term (word), occurs in that document. The second step is the calculation of inverse document frequency (IDF). IDF measures the log of the inverse probability of the term found in all documents [18]. To convert training samples from raw text to TF-IDF vectors, some preprocessing steps are required including tokenization and stop words removal. In this study, performance is evaluated both with and without stop words removal to see the effect of this preprocessing step on the performance of the models, especially in the context of maintenance text. The maximum feature size is fixed to 1024 which is the same size as deep contextualized embedding generated by CamemBERT-large model. Similar to the BOW strategy, the performance of TF-IDF features is also evaluated on training sets with and without augmentation. Furthermore, the same classification methods SVM, DT, and RF are trained. Best models are identified by choosing the best performing models through grid search. The search space for these parameters is presented in Table 4.

Table 4: Range of hyperparameters used for different models to apply grid search

Model	**Hyperparameters for grid search**
SVM *(Support vector machine)*	$C : [1, 10, 100, 1000]$
	$Kernel : [linear, rbf]$
	$gamma : [1e-3, 1e-4]$
DT *(Decision Tree)*	$criterion : [gini, entropy]$
	$max\_depth : [4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 20, 30, 40, 50, 70, 90, 120, 150]$
RF *(Random Forest)*	$bootstrap : [True, False]$
	$max\_features : [auto, sqrt]$
	$min\_samples\_leaf : [1, 2, 4]$
	$min\_samples\_split : [2, 5, 10]$
	$max\_depth : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None]$
	$n\_estimators : [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]$

2.3.5 CamemBERT fine-tuning based models:

Despite low time complexity and simplicity, both BOW and TF-IDF have two major disadvantages. Firstly, with an increase in dataset size, the number of unique words increases, resulting in much larger vectors. Both these methods being frequency-based techniques, vector size is dependent on the number of unique words in the vocabulary. Secondly, these techniques do not handle the order in which the words appear and only focus on the word frequency. That is why it is important to explore recent transformer-based state-of-the-art techniques in NLP like BERT. BERT provides fixed-size contextualized embedding that can be adopted for various downstream tasks, like sequence classification in our case. To adopt the BERT model trained on general language to a specific domain, fine-tuning needs to be performed. To perform

Figure 4: Architecture of domain specific fine-tuning for the BERT based model

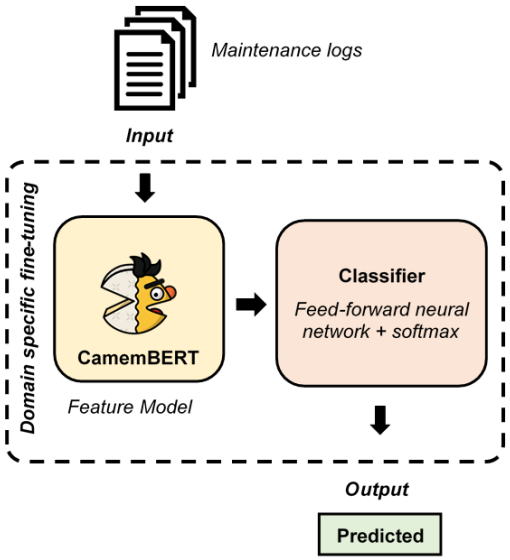

Table 5: Best performance model parameters for CamemBERT fine-tuned models

CamemBERT fine-tuned	**Best Hyperparameters**
Training *(without augmentation)*	$Epochs$: 4 $Batchsize$: 16 $Warmupsteps$: 0 $Optimizer$: $Adam$ ($LearningRate$: 5×10^{-5}) $Maxsequencelength$: 128
Training *(with augmentation)*	$Epochs$: 3 $Batchsize$: 16 $Warmupsteps$: 0 $Optimizer$: $Adam$ ($LearningRate$: 2×10^{-5}) $Maxsequencelength$: 128

domain-specific fine-tuning of BERT-based models, a feed-forward neural network with softmax is jointly trained using pooled BERT output from the pre-trained BERT model. This helps adapt the contextualized embedding previously trained on generalized language data to the target domain using minimal data, time, and computational resources. This approach also requires minimal architecture modification to achieve state-of-the-art performance on the given task. Figure 4 shows the architecture of domain-specific fine-tuning of BERT-based models (CamemBERT in our case).

To train a domain-specific fine-tuned model for free-form maintenance text, CamembertForSequenceClassification class available in *Transformers* Python library was used [21]. This class takes a pre-trained BERT model and the total number of classes as input and adds a feed-forward neural network with a softmax on top of it. Two different versions of this model were trained during system development, one for each training set (with and without augment). Different hyperparameter ranges tested for optimization include: epoch (1, 2, 3 and 4), batch size (8, 16 and 32), max sequence length (64, 96 and 128) and learning rate (2×10^{-5}, 3×10^{-5}, 4×10^{-5} and 5×10^{-5}. The best performance was achieved using hyperparameters presented in Table 5.

2.3.6 CamemBERT feature based models:

In addition to fine-tuning the CamemBERT model on domain-specific data by adding a neural network with a softmax layer on top of it, we can also use the CamemBERT pre-trained model to generate contextualized word embedding. In BERT-based models, the process of converting raw text to the input sequence adds some extra special tokens to the sequence by the inbuilt model tokenizer. One of these special tokens is [CLS]. The embedding of [CLS] token serves as an aggregated representation of the input sequence for classification tasks. These embeddings can be used as features to train some other ML model on top of pre-trained or fine-tuned BERT model [4]. This analysis is performed to compare the effect of contextualized embedding-based features on the same models trained using BOW and TF-IDF-based features. The features obtained from these fine-tuned models consist of fixed size embedding of size 1024. Using embedding from CamemBERT models, other models were trained including SVM, DT, and RF. The hyperparameters of these models are tuned using gird search for provided parameter ranges in table 4.

3 Results and discussion

In this section results of various models trained during this study are compared on the test set. Accuracy and MCC score are presented as performance metrics. Accuracy is the ratio between the correctly predicted instances and all the instances in the test set. Many researchers use accuracy as a standard way to estimate the performance of machine learning systems. The value of accuracy ranges between 0 and 100%, with 0 being the worst and 100% being the best accuracy. Accuracy can be calculated using equation 1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Where TP (true positive) is the number of correctly labelled positive samples, TN (true negative) is the number of correctly labelled negative samples, FP (False Positive) is the number of negative samples incorrectly labelled as positive and FN (false negative) is the number of positive samples incorrectly labelled as negative. MCC score on the other hand is used to provide insight into how well the class imbalance problem was handled. This is because MCC produces a high score only if the prediction obtained good results in all four confusion matrix categories, such as TP, FN, TN, and FP. MCC score ranges between -1 and 1, with -1 being the worst and 1 being the best score. MCC score can be calculated using equation 2.

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TN + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{2}$$

3.1 Performance with and without data augmentation

Various models are trained during the study. Table 6 presents test accuracy and MCC score of various models trained using non augmented and augmented data. Results of the best model are highlighted in the table.

Table 6: Results of various models on test set (training with and without data augmentation)

Features	**Model**	**Non augmented**		**Augmented**	
		Accuracy	**MCC**[6]	**Accuracy**	**MCC**
BoW[1]	SVM[3]	59.76	0.43	60.98	0.46
	DT[4]	58.54	0.42	52.44	0.36
	RF[5]	71.95	0.61	64.63	0.50
TF-IDF[2] *(with stopwords)*	SVM	70.73	0.58	65.85	0.53
	DT	52.44	0.32	51.22	0.32
	RF	69.51	0.57	70.73	0.59
TF-IDF *(without stopwords)*	SVM	65.85	0.51	64.63	0.50
	DT	57.32	0.40	60.98	0.45
	RF	68.29	0.55	67.07	0.54
CamemBERT Embedding	SVM	73.17	0.63	78.05	0.70
	DT	70.73	0.59	69.51	0.57
	RF	74.39	0.65	76.83	0.68
Preprocessed Text	CamemBERT-Large *with* FFNN[7] + Softmax ↓ *CamemBERT fine-tuned*	**80.21**	**0.69**	**82.29**	**0.71**

[1] *BoW: Bag of Words,* [2] *TF-IDF: Term Frequency - Inverse Document Frequency*

[3] *SVM: Support Vector Machine,* [4] *DT: Decision Tree,* [5] *RF: Random Forest*

[6] *MCC: Matthews Correlation Coefficient,* [7] *FFNN: Feed-forward Neural Network*

Results indicate that the CamemBERT model with fine-tuning approach has the best performance with respect to both accuracy and MCC score. It can also be observed that all the models are performing better using contextualized embedding compared to other features, both in terms of accuracy and MCC score.

Similarly, table 6 also presents the accuracy and MCC score of various models after data augmentation. Results indicate an increase in the accuracy and MCC score for the CamemBERT model with fine-tuning approach after data augmentation. However, in both augmented and non augmented data cases CamemBERT with fine-tuning approach is the best performing model.

In summary results show that deep contextualized embedding can be used to develop ML applications using free-form text. Although results of the CamemBERT model with fine-tuning approach (best model) were acceptable without data augmentation, results indicate that data augmentation helps to reduce the effect of class imbalance. Furthermore, deep contextualized embedding when used as a feature in classical machine learning algorithms (SVM, RF, and DT) also produce better results compared to classical feature techniques such as BOW and TF-IDF.

4 Conclusion

This study explored the capability of the state-of-the-art BERT-based French NLP model, CamemBERT on free-form maintenance text. The purpose of the study was to estimate the category of maintenance problems using log entries by maintenance operators. CamemBERT was used to categorize the text using domain-specific transfer learning both in fine-tuning and feature-based mode. The dataset used for the study is a real industrial dataset. The provided data set contained highly unstructured text with class imbalance; as some problem categories are more common compared to others. To handle the class imbalance problem data augmentation was performed using word-level random substitution with CamemBERT based contextualized word embedding. Findings indicate that CamemBERT with a fine-tuning approach resulted in the best accuracy and MCC score on the test set using both training sets (with and without augmentation). Similarly, it can be observed that other models (SVM, DT, RF) when trained using contextualized embedding-based features gave better performance compared to classical features like BOW and TF-IDF. The best model was able to achieve test accuracy of 80.21% with an MCC of 0.69 when trained using the training set without augmentation and accuracy of 82.29% and MCC of 0.71 when trained using the training set with augmentation. This indicates data augmentation helps increase model performance by handling class imbalance. The study also shows using BERT-based state-of-the-art models we can effectively explore the text in maintenance logs to develop applications for industrial processes improvement.

5 Future work

In the context of the Technical language Processing (TLP) paradigm and industry 4.0 [3], there is a need for similar studies to adopt NLP pipelines for free-form maintenance text. In continuation with this study future work will focus on the following directions:

- This study is the initial work towards the development of maintenance applications based on free-form maintenance text to facilitate engineers to efficiently perform maintenance tasks.
- Future work will also include the comparison between the performance of similar French BERT-based models like FlauBERT to analyze which model performs the best on free-form maintenance text.
- Performance analysis on other similar industrial datasets to identify generalized approaches to handle this type of data. Also, various data augmentation techniques need to be explored and tested on multiple datasets to identify the best approach for free-form form maintenance text.

6 Acknowledgments

This work has been supported by the EIPHI Graduate school (contract "ANR-17-EURE-0002").

References

[1] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, 2016.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[3] Michael P Brundage, Thurston Sexton, Melinda Hodkiewicz, Alden Dima, and Sarah Lukens. Technical language processing: Unlocking maintenance knowledge. *Manufacturing Letters*, 27:42–46, 2021.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*, 2021.

[6] Dennis D Hirsch. The glass house effect: Big data, the new oil, and the power of analogy. *Me. L. Rev.*, 66:373, 2013.

[7] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[8] Edward Ma. Nlp augmentation. https://github.com/makcedward/nlpaug, 2019.

[9] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. Camembert: a tasty french language model. *arXiv preprint arXiv:1911.03894*, 2019.

[10] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. *arXiv preprint arXiv:1708.00107*, 2017.

[11] Yash Mehta, Navonil Majumder, Alexander Gelbukh, and Erik Cambria. Recent trends in deep learning based personality detection. *Artificial Intelligence Review*, pages 1–27, 2019.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[13] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[14] Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*, 2017.

[15] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[16] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[17] Sebastian Ruder. *Neural transfer learning for natural language processing*. PhD thesis, NUI Galway, 2019.

[18] Xia Tian and Wang Tong. An improvement to tf: term distribution based term weight algorithm. In *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, volume 1, pages 252–255. IEEE, 2010.

[19] Juan Pablo Usuga Cadavid, Bernard Grabot, Samir Lamouri, Robert Pellerin, and Arnaud Fortin. Valuing free-form text data from maintenance logs through transfer learning with camembert. *Enterprise Information Systems*, pages 1–29, 2020.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[21] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.