CBR-Based Decision Support System for Maintenance Text Using NLP for an Aviation Case Study

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

[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

Recently, Prognostics and Health Management (PHM) has emerged to promote predictive maintenance as a methodological key to overcome the limitations of traditional reliability analysis. The Natural Language Processing (NLP) methods allow the maintenance log usage for maintenance diagnostics and decision making. The Maintenance Work Orders (MWOs) contain vital health indicators and decades of experience related to various maintenance actions. However, due to the unstructured nature of maintenance text, it is not common to develop a tool using these textual maintenance entries. This paper proposes a textual Case-Based Reasoning (CBR) approach combined with Technical Language Processing (TLP) to find solutions for new problems based on previous experiences. The Bidirectional Encoder Representations from Transformers (BERT) model is adopted for maintenance data using unsupervised fine-tuning technique Transformer-based Sequential Denoising Auto-Encoder (TSDAE) for aviation case study. Results show that the pre-trained BERT model can adopt domain-specific data and produce semantic matches with only a small amount (1000 samples) of domain specific data.

Keywords— Prognostics and Health Management (PHM), Case-Based Reasoning, Natural Language Processing, Technical Language Processing, BERT, Sequential Denoising Auto-Encoder

1 Introduction

Prognostic and Health Management (PHM) represents one of the most advanced maintenance strategies available today. PHM monitors the equipment health assessment, performs the diagnostics and prognostics, and provides design rules for maintenance [17]. Many works are interested in PHM by interpreting condition monitoring signals such as vibration, acoustic emission, and pressure [12]. However, few of them are focused on maintenance logs for diagnostics and decision making. Maintenance reports and logs represent the essential data and information generated by maintenance activities [24]. The maintenance log consists of the operators' or technicians' notes on the problems encountered, and the solutions applied. However, human-generated data is unstructured, often inconsistent and ambiguous, contains errors and is filled with domain-specific jargon and abbreviations [15]. A database of maintenance logs can provide problem descriptions, causes and solutions that appeared previously for a system. However, different technicians rarely describe the same problem when similar problems occur in the same way [23]. This leads to inconsistencies in the description in the database, making it challenging to categorize problems or learn about similar causes.

In maintenance, the problems are recurrent. Thus previously documented solutions can be successfully reused. Different methodologies can be implemented for a given diagnostic domain. One of the most

appropriate is Case-Based Reasoning (CBR). CBR is a nature-inspired problem-solving methodology. CBR is inspired by nature's ability to learn continuously from experience [14]. Each newly solved problem and its corresponding solution are stored in its central repository of knowledge called a case base. The first working principle of CBR is that similar problems have similar solutions [10]. Thus, the existing cases and their solutions from the case repository are used to solve a new case by finding cases similar to a target problem [2]. CBR is an excellent tool for reusing previously acquired experience and is widely used to production machine detection [16] [13], cost optimization [14] [11] or in fault diagnostics [4].

Textual Case-Based Reasoning (TCBR) is a CBR sub field that compares a problem with a set of past cases where the knowledge sources are available in textual format. Comparing text content is vital to identify relevant cases for solution reuse [9]. However, Textual information typically is hard to process by computers due to its relatively unstructured format and the numerous possible variations in its interpretation. Hence, it is often necessary to introduce additional processing to extract structured knowledge from textual documents [22].

In fact, TCBR has been adopted to solve problems using textual data for maintenance. Zhong et al. [28] have presented in 2018 a text CBR framework that uses the diagnostics ontology to annotate fault features recorded in the repair verbatim. The case retrieval is employed to search for the best-practice repair actions for fixing faulty parts. Dendani et al. [6] have developed a CBR application for steam turbine fault diagnostics that integrates domain knowledge modelling in ontological form.

Even though there are many papers concerning CBR and textual data, only very few papers consider its application with Natural Language Processing (NLP) and special for unstructured maintenance text. Using NLP for maintenance text needs to be adapted to understand and meet the requirements of technical data. Thus, Brundage et al. [3] have proposed a Technical Language Processing (TLP) pipeline for technical engineering textual data. Naqvi et al. [18] have proposed a method to classify maintenance records into various categories (mechanical, electronic, electrical, etc.) using BERT to analyze the performance of BERT on maintenance text. TLP uses technical data that considers industrial engineering use cases as raw text input for an NLP method [8]. The TLP aims to improve support tools to reduce errors and increase confidence in text analysis through collaboration between analysts and domain experts [19].

This paper combines CBR with TLP using the BERT model to realize the information retrieval from maintenance logs. The proposed approach will assist maintainers in deciding by looking for a similar problem and solution. The rest of this paper is organized as follows: Section 2 introduces the problem statement and the description of the case study; Section 3 explains the overall proposed approach of the CBR and TLP using BERT model; Section 4 focuses on the results and discussion; The final section contains a conclusion and suggestions for future work.

2 Aviation case study

In this section, an aviation case study is discussed in detail, including description of the problem statement, dataset used for the case study and hypothesis.

2.1 Problem statement

Across various industries, it is a common practice to record maintenance activities in case of a breakdown. These maintenance entries recorded over time contain vital health indicators for different asserts. In addition to these health indicator maintenance entries also contain decades of experience related to the various maintenance actions to solve different problems. Unfortunately due to the unstructured nature of maintenance text, it is not a common practice to develop a tool using these textual maintenance entries. Thus not contributing to the development of tools to help improve the maintenance process.

In this case study, we propose an aviation maintenance decision support system based on CBR using maintenance text. The developed system will be able to help train new maintenance operators with less field experience and will also help the experienced operator to query past knowledge to find quicker solutions. This system will also help to combine past and new maintenance records into a single knowledge base. In case of a new problem, the operator can input the problem query into the system and can find similar problems previously experienced along with relevant actions to perform. The decision support is based on the CBR framework that evolve and improve with the addition of new cases to the knowledge base.

In the past few years, there has been a huge performance improvement in the field of NLP [21, 25]. Special the introduction of pre-trained models in the field of NLP has transformed the way text used to be processed [27]. Currently, there are only a few studies that explore the capability of state-of-the-art pre-trained NLP models on different types of textual information specially the maintenance text. Our hypothesis states that with proper fine-tuning these pre-trained models trained on 100 GB's of language data should be able to adopt maintenance text. Eventually, the fine-tuned model can be used as similarity model to measure maintenance text similarity between cases.

2.2 Aviation dataset

The dataset used for the case study consists of 500 random samples of aviation maintenance dataset. The dataset is from the University of North Dakota Aviation Program. It is available on the MaintNet[1] website to encourage the research on maintenance text [1]. The shared dataset consists of three columns. The first column is the case identifier (case id). The second column is the description of the problem (case description). Finally, the third column is the action performed to resolve the problem or solution to the problem. Table 1 shows some randomly selected samples from the dataset used in the case study.

Table 1: Random samples from Maintnet's aviation dataset.

Id	**Problem**	**Action**
100067	FUEL PRESS 32 - 33 W/ JUMPS INTO RED	CLEANED TRANSDUCER GROUND PATH. FUEL PRESS CHECKED GOOD
100108	#2 & 4 CYL OIL RETURN LINE CLAMPS LOOSE	TIGHTENED CLAMPS
100119	#1 & 3 ROCKER COVER GASKETS ARE LEAKING	REMOVED & REPLACED GASKETS
100151	ENGINE OVERSPEED UP TO 3400 RPM @ TERMINATION OF AN AUTO.	BEGAN ENG OVERSPEED INSP IAW . IDENTIFIED & CORR

The unstructured and unconventional nature of maintenance text can be observed in the samples from Table 1. These maintenance logs are different from the regular text in terms of sentence structure and usage of words. To test the performance for this case study, we used BERT with minimum pre-processing. Although the version of the BERT model used for this study is not case sensitive, we converted all the maintenance logs to lower case for normalization and better visualization.

[1] `https://people.rit.edu/fa3019/MaintNet/`

3 Proposed Methodology

This section is dedicated to discussion about the development of a decision support system using the CBR framework. This section explains various aspects of the CBR framework including the similarity model, knowledge base and CBR cycle. Figure 1 shows the flow diagram of CBR framework.

3.1 CBR framework

The first step of the CBR cycle is retrieval, where we match the new problem case to the past cases. To find similar cases for the new problem case we first need to process the text and convert it into the embedding (numerical features). There are different methods to achieve this transformation including classical techniques such as BOW (Bag-of-Words), TF-IDF (Term Frequency – Inverse Document Frequency), and more recent techniques involving large pre-trained language models. The recent advancements in the field of computational linguistics shifted the focus from building task specific models from scratch to general-purpose pre-trained language models. One of the most commonly used pre-trained language models is BERT (Bidirectional Encoder Representations from Transformers) [5]. There are many BERT-based pre-trained models available trained using different datasets and configurations. Depending on the variant of the BERT model (base or large) final feature vector size is either 768 or 1024 respectively. In this case study, we used the BERT-base-uncased model that was released with the original paper [7] and is available on hugging face website[2]. Model BERT-base-uncased has a total 12-layer with 768-hidden units in each later consisting total of 110 million parameters. The size of the final feature embedding of this model is also 768 which is equal to the size of hidden units. As the name states the model is uncased which means that it is not case-sensitive. In this case study, the performance of the BERT-base-uncased model is compared before and after the domain-specific fine-tuning model. The idea is to analyze the performance gain after domain-specific fine-tuning and to see how well these models can handle unstructured maintenance text with and without fine-tuning. The fine-tuning technique used for this case study is discussed in the following.

3.1.1 Domain fine-tuning of BERT model

To use BERT for feature extraction, we can either directly use a pre-trained model or we can further fine-tune the model on domain-specific data. This way pre-trained model can better adapt to domain-specific data, which eventually contributes to improved performance. Also, for the semantic similarity task, it is important to fine-tune the model since pre-trained BERT does not produce a good sentence or paragraph-level embedding [20]. To resolve this problem sentence-transformers[3] library was introduced to serve as a resource for techniques to adopt transformer-based models for sentence or paragraph level embedding.

Depending on the type of data and the underlying task fine-tuning can be performed using supervised and unsupervised methods. For this case study as the data is not labeled we used the unsupervised fine-tuning technique Transformer-based Sequential Denoising Auto-Encoder (TSDAE) [26]. TSDAE takes a pre-trained transformer-based model and fine-tunes the models using a sequential denoising auto-encoder using only sentences from the domain-specific data without any labels. The output of the process is a domain fine-tuned sentence BERT model. The resulting model gives better sentence and paragraph level embedding compared to the pre-trained version of the input model. During the unsupervised training process, TSDAE first encodes the noisy sentences into a fixed-length sentence embedding vector and then uses a decoder to reconstruct the original sentence using the generated sentence embedding. After the training process, the trained encoder is used to generate final sentence embedding. In this case study, we used 1000 input sentences (500 problems + 500 actions) in the fine-tuning process. The size of the final sentence embedding vector

[2] `https://huggingface.co/bert-base-uncased`
[3] `https://www.sbert.net/`

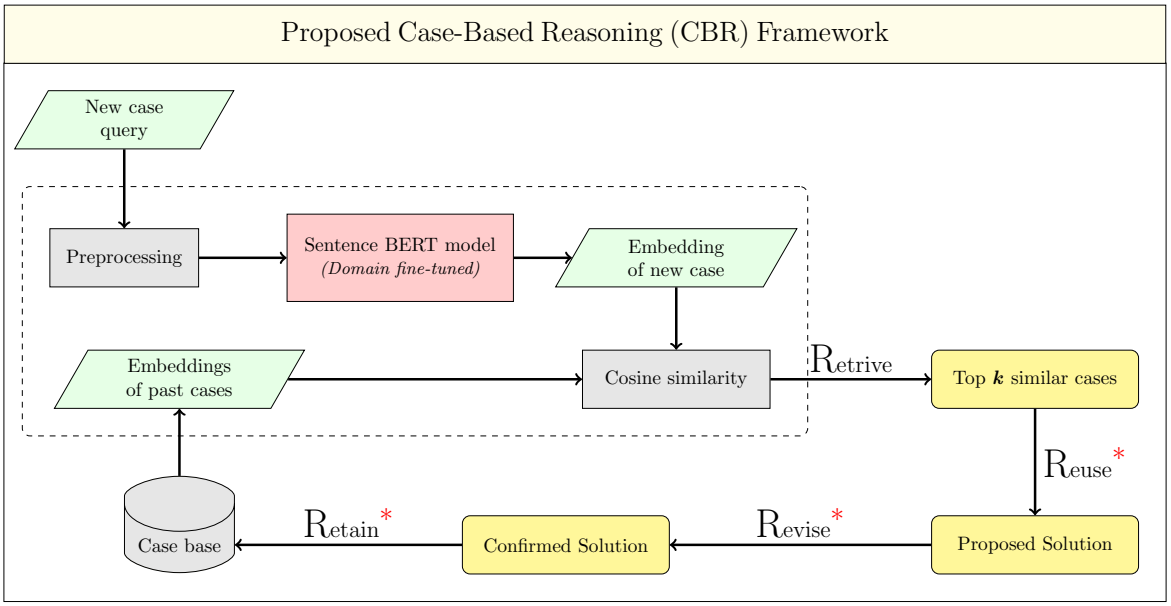

Figure 1: Case-Based Reasoning (CBR) Framework

is 768 that is the embedding size of the BERT-base-uncased model. TSDAE currently has state-of-the-art results compared to other unsupervised fine-tuning techniques.

3.1.2 Knowledge base

One of the important components of a CBR system is the knowledge base where all the information about the past cases is stored. We are storing embeddings of past cases in the knowledge base with the textual information. It is important because it reduces the retrieval time by avoiding regeneration of embeddings for each search. Instead, we only use the model to generate embedding for new problem queries and compare it with pre-generated past case embeddings.

3.1.3 CBR cycle - Retrieval

In the CBR cycle when a new problem query is exposed to the CBR system the first step is to retrieve similar cases. This retrieval process has various steps. First, pre-process the input problem query. In our case, we are only changing the case of the text to the lower case as the only pre-processing step. After pre-processing the new problem query text is passed as input to the domain fine-tuned sentence embedding model to generate its embedding. Then this newly generated embedding is compared with the embeddings of all the past cases in the knowledge base. As these embeddings are dense feature vectors, we used cosine similarity to calculate the similarity between these feature vectors. Equation 1 shows formulation for cosine similarity which is the measure of similarity between two non-zero vectors:

$$cosine\ similarity = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \times \|\mathbf{B}\|} \tag{1}$$

In cosine similarity, we calculate the cosine of the angle between input vectors which dot product of input vectors divided by the product of lengths of these input vectors. Cosine similarity does not focus on the magnitude but on the angle between compared vectors. Its values range between -1 and 1, where cosine similarity is -1 between two opposite vectors, 0 between two orthogonal vectors, and 1 if the compared vectors are proportional. If the cosine similarity is used in a positive space the outcome is bounded between 0 and 1, where 0 represents minimum or no similarity and 1 represents maximum similarity. In our case study cosine similarity value also ranges between 0 and 1. The output of the retrieval step is top k potential solution cases for the given problem query where k is the number of similar cases required during the retrieval phase.

3.1.4 CBR cycle - Reuse, Revise and Retain

After retrieval of top k similar cases, the maintenance operator goes through these cases and tries to find the most relevant case to solve the new problem. After finding the most relevant case operator tries to apply this case to solve the new problem. This step is called reuse in the CBR cycle. The next step in the CBR cycle is to revise. Sometimes operators have to make revisions to the proposed solution because it is not fully appropriate to solve the problem or the operator does not find the relevant solution to the new problem query. After revising the solution and solving the new problem operator has the confirmed solution to the new problem. After this confirmation, the operator proceeds to save the new problem query and the confirmed solution to the knowledge base. This evolutionary process helps the CBR system to get better at solving new problems.

4 Results and discussion

In this section, we discussed the results of different models used for the case study to analyze the performance of the CBR-based decision support system. We presented two different types of results: precision-based results and semantic similarity-based results.

4.1 Evaluation process

First, we will discuss precision-based results. As mentioned in Section 3.1 we can use BERT to generate features from the text in the pre-trained as well as the fine-tuned state. To analyze the performance boost due to domain-specific fine-tuning, Table 2 shows the results of precision for both pre-trained and fine-tuned models. The dataset used for the case study is unlabeled, so to calculate performance metrics we developed a set of 17 problem queries by analyzing past cases in the knowledge base. These queries are treated as sample problems to test the performance of the developed CBR-based decision support system. For each problem query, we extracted top k similar cases in cosine similarity score after comparing the feature vector of the query with features vectors of past cases. For this analysis value of k is set to 5, the reason behind the selection of this number is the availability of at least 5 similar cases in the knowledge base for each query in the query set. To calculate precision we asked maintenance experts to individually compare extracted similar cases against the input problem query. If the input problem query and the predicted top similarity case is identical, we considered it as True Positive (TP), and if not we considered it as False Positive (FP). Finally, based on the count of true positive and false positive cases out of the top 5 identified cases we calculated the precision for each query. We also calculated the average precision of the problem query set for the pre-trained as well as fine-tuned model. Equation 2 shows the formula to calculate precision which represents the proportion of correct identifications:

$$Precision = \frac{True\ Positive}{True\ Positive\ +\ False\ Positive} \tag{2}$$

We also analyzed similar cases identified by the best-performing model (fine-tuned model) for exact or semantic matches. To generate these annotation-based results, we first set a similarity threshold. Threshold ensures that we only pick cases that have a certain level of similarity to the query cases instead of picking a fixed number of cases that may or may not be relevant. After getting similar cases from the model, we only selected unique cases to remove repetitions. Afterward, we manually reviewed predicted similar cases and assigned an exact match when a predicted similar case and problem query share a similar structure or words. A semantic match is when a predicted similar case and the problem query are semantically similar but contain different structure or words.

4.2 Results

Table 2 shows the precision based results for problem query test set.

Table 2: Precision Comparison Between Pre-trained and Fine-tuned Model

		Precision (%)	
Id	**Query**	***pre-trained***	***fine-tuned***
1	engine running rough	40	100
2	engine killed	80	100
3	engine overspeed	80	80
4	engine vibrating	0	100
5	leakage	100	100
6	screw loose	0	100
7	screw missing	20	100
8	baffle cracked	20	80
9	remove engine	0	60
10	baffle plug	40	100
11	intake leaking	100	100
12	broken rivet	0	100
13	firewall repair	100	100
14	engine torquemeter	100	100
15	gasket leaking	100	100
16	rocker cover loose	100	100
17	loose clamp	100	100
	Average precision	**58**	**95**
	Total 5/5 TP[1] cases	**7 out of 17**	**14 out of 17**
	Total 5/5 FP[2] cases	**4 out of 17**	**0 out of 17**

[1] ***TP:*** *True positive,* [2] **FP:** False positive

For better understanding, we scaled the precision to be represented as percentage. Results indicate, average precision of 58% and 95% for the pre-trained model and fine-tuned model respectively. Similarly pre-trained model has 7 out of 17, 5/5 TP cases while the fine-tuned model has 14 out of 17, 5/5 TP cases.

Furthermore fine-tuned model does not have any 5/5 FP cases while the pre-trained model has 4 out of 17 FP cases. It is important to note after the fine-tuning model was able to identify relevant cases in most for most of the sample problems from the query set. Also after fine-tuning the model does not have any 5/5 FP cases.

Annotation-based results for best model (fine-tuned model) are presented in Tables 3 and 4.

Table 3: Annotation based analysis of semantic similarity for query 2

Exact match	**Semantic match**	
Query: engine killed		**Similarity**
Similar case 1: engine idle override killed engine.		0.79
Similar case 2: while taxiing, engine quit		0.77
Similar case 3: engine died during mag ck.		0.72
Similar case 4: engine would not start.		0.69
Similar case 5: engine runs rough after start. engine shut down when power t.		0.66

For query 2 after setting cosine similarity threshold at 0.6, we got multiple similar cases. Table 3 shows 5 similar cases for query 2 including both semantically similar responses and exact matches. For example, in similar case 1, we got exact match *"killed engine"*. While in similar case 2, 3, 4 and 5 we got semantically similar matches like *"engine quit"*, *"engine died"*, *"engine would not start"* and *"engine shut down"*. In Table 3 we also presented the cosine similarity value of query cases with the predicted similar cases.

Table 4: Annotation based analysis of semantic similarity for query 13

Exact match	**Semantic match**	
Query: broken rivet		**Similarity**
Similar case 1: aft baffle bracket rivets sheared.		0.67
Similar case 2: rivet pulled through baffle seal, #1 cyl.		0.67
Similar case 3: r/h side back baffle is cracked & bracket rivets broken.		0.65
Similar case 4: top front r/h baffle, baffle seal rivet pulled through.		0.63

Similarly in Table 4 it can be observed that given query *"broken rivet"* and after setting cosine similarity

threshold at 0.6, we got 4 similar cases. These cases also include both semantically similar responses and exact matches. For example, in similar case 1, 2 and 4 we got semantically similar matches like *"rivets sheared"*, *"rivet pulled through baffle seal"* and *"baffle seal rivet pulled through"*. While in similar case 3, we got exact match *"rivets broken"* same as query. In Table 4 we also presented the cosine similarity value of query cases with the predicted similar cases.

5 Conclusion and Future Work

In this case study, we developed a CBR-based decision support system for aviation maintenance use case. Results indicate that the recent state-of-the-art models like BERT can be used to process maintenance text with proper fine-tuning without the need for a special preprocessing pipeline. It is also important to note that with only 1000 sentences pre-trained BERT model can adopt domain-specific data and produce semantic matches. We also compared the performance of the pre-trained model with the fine-tuned model to observe the performance gain after domain-specific fine-tuning. Results indicate an improvement of 37% in the precision of fine-tuned model compared to the pre-trained model. The developed system can be helpful to the new hires (with less experience) to identify solutions to various problems without extensive supervision. Also, the developed CBR based decision support system serves as a knowledge base of the past maintenance knowledge for the company and can be further utilized to improve the maintenance process.

Some of the future work planned for this case study includes: (i) developing new metrics for detailed performance analysis; (ii) identifying new ways to measure the quality of semantic similarity for predictions; (iii) analysing performance of the developed framework on other unsupervised fine-tuning approaches; (iv) testing the performance of the developed framework on other transformer-based models such as BERT-large, XLNet and others.

Acknowledgment

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

References

[1] Farhad Akhbardeh, Travis Desell, and Marcos Zampieri. Maintnet: A collaborative open-source library for predictive maintenance language resources. *arXiv preprint arXiv:2005.12443*, 2020.

[2] Safa Ben Ayed, Zied Elouedi, and Eric Lefevre. Detd: dynamic policy for case base maintenance based on ek-nnclus algorithm and case types detection. In *International conference on information processing and management of uncertainty in knowledge-based systems*, pages 370–382. Springer, 2018.

[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] Sumana De and Baisakhi Chakraborty. Case based reasoning (cbr) methodology for car fault diagnosis system (cfds) using decision tree and jaccard similarity method. In *2018 3rd International Conference for Convergence in Technology (I2CT)*, pages 1–6. IEEE, 2018.

[5] Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. Bertje: A dutch bert model. *arXiv preprint arXiv:1912.09582*, 2019.

[6] Nadjette Dendani-Hadiby and Mohamed Tarek Khadir. A case based reasoning system based on domain ontology for fault diagnosis of steam turbines. *International Journal of Hybrid Information Technology*, 5(3):89–104, 2012.

[7] 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.

[8] Alden Dima, Sarah Lukens, Melinda Hodkiewicz, Thurston Sexton, and Michael P Brundage. Adapting natural language processing for technical text. *Applied AI Letters*, 2(3):e33, 2021.

[9] Islam Elhalwany, Ammar Mohammed, Khaled T Wassif, and Hesham A Hefny. Enhancements to knowledge discovery framework of sophia textual case-based reasoning. *Egyptian Informatics Journal*, 15(3):211–220, 2014.

[10] Zhi-Ping Fan, Yong-Hai Li, Xiaohuan Wang, and Yang Liu. Hybrid similarity measure for case retrieval in cbr and its application to emergency response towards gas explosion. *Expert Systems with Applications*, 41(5):2526–2534, 2014.

[11] Alfonso González-Briones, Javier Prieto, Fernando De La Prieta, Enrique Herrera-Viedma, and Juan M Corchado. Energy optimization using a case-based reasoning strategy. *Sensors*, 18(3):865, 2018.

[12] Kamran Javed, Rafael Gouriveau, Xiang Li, and Noureddine Zerhouni. Tool wear monitoring and prognostics challenges: a comparison of connectionist methods toward an adaptive ensemble model. *Journal of Intelligent Manufacturing*, 29(8):1873–1890, 2018.

[13] Mohammad Reza Khosravani, Sara Nasiri, and Kerstin Weinberg. Application of case-based reasoning in a fault detection system on production of drippers. *Applied Soft Computing*, 75:227–232, 2019.

[14] Nahyun Kwon, Kwonsik Song, Yonghan Ahn, Moonsun Park, and Youjin Jang. Maintenance cost prediction for aging residential buildings based on case-based reasoning and genetic algorithm. *Journal of Building Engineering*, 28:101006, 2020.

[15] Zhibin Li, Jian Zhang, Qiang Wu, and Christina Kirsch. Field-regularised factorization machines for mining the maintenance logs of equipment. In *Australasian Joint Conference on Artificial Intelligence*, pages 172–183. Springer, 2018.

[16] Marisa Marisa, Suhadi Suhadi, Muhamad Nur, Prima Dina Atika, Sugiyatno Sugiyatno, and Davi Afandi. Factory production machine damage detection system using case-based reasoning method. In *2021 8th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, pages 374–380. IEEE, 2021.

[17] Safa Meraghni, Labib Sadek Terrissa, Meiling Yue, Jian Ma, Samir Jemei, and Noureddine Zerhouni. A data-driven digital-twin prognostics method for proton exchange membrane fuel cell remaining useful life prediction. *International Journal of Hydrogen Energy*, 46(2):2555–2564, 2021.

[18] Syed Meesam Raza Naqvi, Christophe Varnier, Jean-Marc Nicod, Noureddine Zerhouni, and Mohammad Ghufran. Leveraging free-form text in maintenance logs through bert transfer learning. In *International Conference on Deep Learning, Artificial Intelligence and Robotics*, pages 63–75. Springer, 2022.

[19] Maria Vatshaug Ottermo, Solfrid Håbrekke, Stein Hauge, and Lars Bodsberg. Technical language processing for efficient classification of failure events for safety critical equipment. In *PHM Society European Conference*, volume 6, pages 9–9, 2021.

[20] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

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

[22] Spencer Rugaber, Shruti Bhati, Vedanuj Goswami, Evangelia Spiliopoulou, Sasha Azad, Sridevi Koushik, Rishikesh Kulkarni, Mithun Kumble, Sriya Sarathy, and Ashok Goel. Knowledge extraction and annotation for cross-domain textual case-based reasoning in biologically inspired design. In *International Conference on Case-Based Reasoning*, pages 342–355. Springer, 2016.

[23] Thurston Sexton, Melinda Hodkiewicz, Michael P Brundage, and Thomas Smoker. Benchmarking for keyword extraction methodologies in maintenance work orders. In *PHM society conference*, volume 10, 2018.

[24] 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.

[25] 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.

[26] Kexin Wang, Nils Reimers, and Iryna Gurevych. Tsdae: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning. *arXiv preprint arXiv:2104.06979*, 2021.

[27] 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.

[28] Zhiwang Zhong, Tianhua Xu, Feng Wang, and Tao Tang. Text case-based reasoning framework for fault diagnosis and predication by cloud computing. *Mathematical Problems in Engineering*, 2018, 2018.